$ 7,95

GAMES for the ATARI®

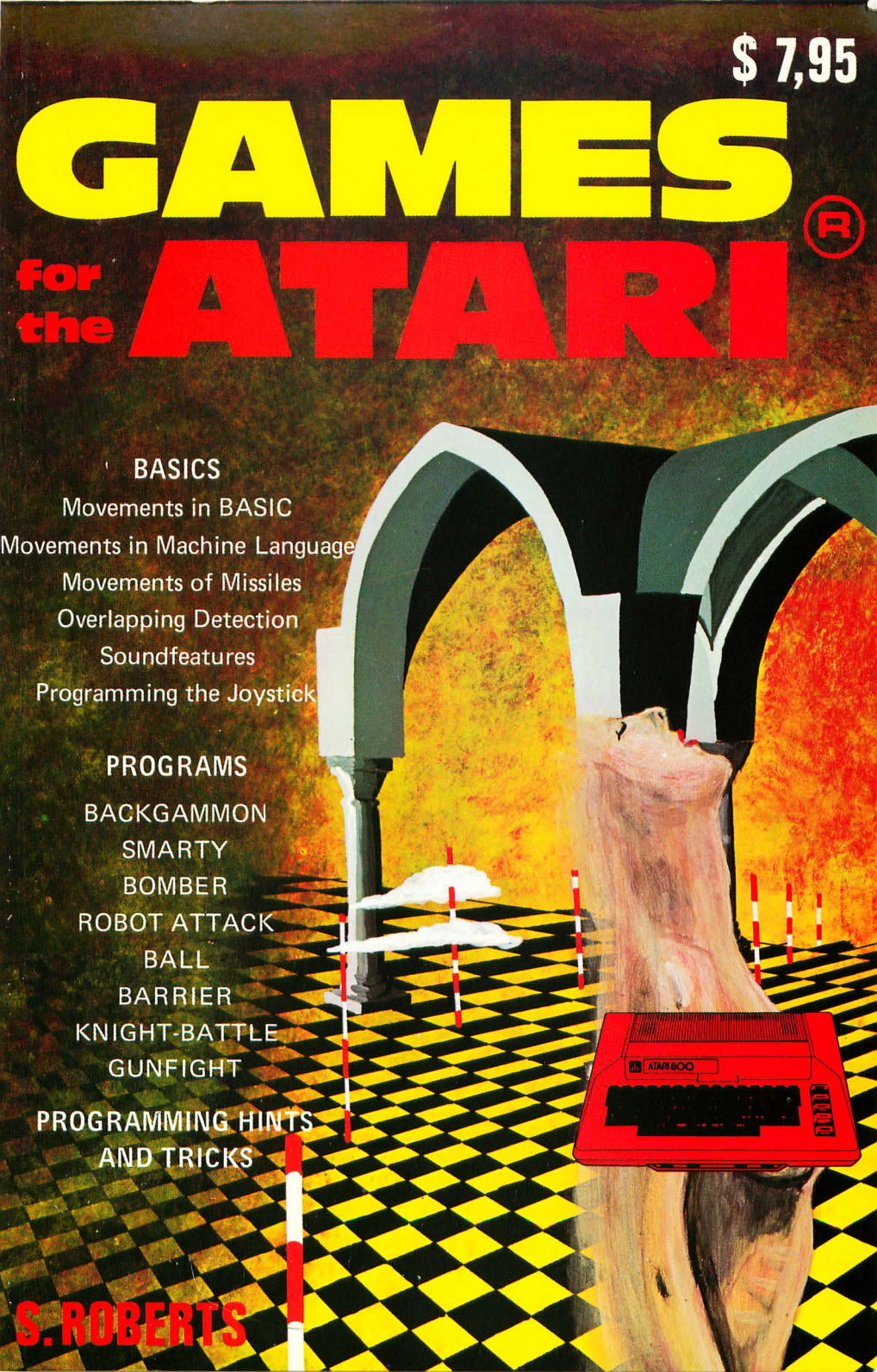
BASICS

Movements in BASIC
Movements in Machine Language
Movements of Missiles
Overlapping Detection
Soundfeatures
Programming the Joystick

PROGRAMS

BACKGAMMON
SMARTY
BOMBER
ROBOT ATTACK
BALL
BARRIER
KNIGHT-BATTLE
GUNFIGHT

PROGRAMMING HINTS AND TRICKS



S. ROBERTS

This book is an independent production of Ing. W. HOFACKER GMBH International. It is published as a service to all ATARI personal computer users worldwide.
All rights reserved. No Part of this book may be reproduced by any means without the express written permission of the publisher. Example programs are for personal use only. Every reasonable effort has been made to ensure accuracy throughout this book, but neither the author or publisher can assume responsibility for any errors or omissions. No liability is assumed for any direct, or indirect, damages resulting from the use of information contained herein.
First Edition
First Printing
May 1982 in the Federal Republic of Germany
© Copyright by Winfried Hofacker
Cover Design: H.-F. Muenchberg
W. Hofacker

Acknowledgement

Thanks to Ekkehard Floegel
Franz Ende
John Kozero
Robert Lloyd
Christoph Goethe
Hans-Christoph Wagner
Günther Eibel
H. Hannenberg
Sefie Maier
for their help in completing this book.

ISBN 3-911682 - 84 - 3

* Reference is made to ATARI throughout this book. ATARI is a trademark of ATARI Inc., a division of Warner Communications Company.

Publisher:
Ing. W. HOFACKER GmbH, Tegernseerstr. 18, D-8150 Holzkirchen,W.-Germany

GAMES FOR THE ATARI 400/800

OR

HOW TO PROGRAM YOUR OWN GAMES

PREFACE

Among the many microcomputers on the market today some are designed more for scientific or control purposes while others are more suitable for great graphics and games. The ATARI microcomputers 400 and 800 belong to the second group.

The ATARI computers provide many possibilities to create on-the-screen-graphics and also to produce almost any sound effect.

The two features – graphics and sound – are exactly what's needed to program computer games.

This booklet provides ideas on how to create your own computer games.
This booklet deals primarily with BASIC examples. However, for very advanced programs, BASIC may be too slow; consequently, such programs should be written in machine language. But since not everybody knows how to program a 6502 microprocessor, we have included only one example in machine language at the end of the book.

Important Notice

A HINT:
Users of ATARI-computers with only 16K of RAM may have problems with programs that use a character list in the RAM area. For example the following program:

```
100 FOR I=0 TO 1023
110 POKE 16384+I, PEEK(57344+I)
120 NEXT I
130 POKE 756,64
```

This program transfers the character list (1K long) to $4000, but a computer with only 16K doesn' t have any memory at that location. So we have to transfer to $2000=8192.
In location 756 there has to be the page boundary where the character list is to be found. If this is $2000, there has to be the number 32 in that location. The display-list also is at a different location then. It starts at 15392 instead of 39968.

TABLE OF CONTENTS

Drawing Figures on the Screen 001
Movements in BASIC 007
Movements in Machine Language 009
Movements of Missiles 012
Overlapping Detection 014
Soundfeatures 016
Programming the Joystick 018
Backgammon 019
SMARTY 028
BOMBER 033
ROBOT ATTACK 038
BALL 042
SMART 044
BARRIER 049
KNIGHT-BATTLE 055
CALENDAR 061
GUNFIGHT 063
Appendix 091
The Video Processor "ANTIC" and the ATARI 400/800 .. 096
Display List Interrupts and the ATARI 101
ATARI 400/800 and CTIA / GTIA 109
The ATARI 400/800 and its Character Set 111

Useful commands for programming games

Graphics commands

GRAPHICS n
SETCOLOR I,m,n
COLOR n
PLOT x,y
DRAWTO x,y

Sound commands

SOUND v,u,t,I (4 independent channels)

Additional commands

Changing the hardware registers by PEEK and POKE

Important Notice

The programs in this book were developed and tested on the ATARI 800/48K RAM. If you want to run the programs on other machines, please check, if there is enough memory.
Some programs for example transfer the character set to hex 4000. The ATARI 400/16K has no RAM in this area. So you have to make the corresponding adjustments.
If you put the character set into $ 2000 = 8192 dec. you can also use a 16K machine. Don' t forget to set the boundary of the character set at location 756 dec. If you use $ 2000, location 756 must be 32 dec.
Please check each program and change the appropriate locations.

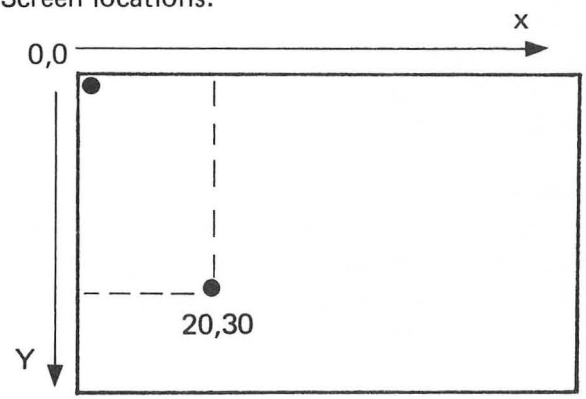
Colors may be different on different ATARI computers (PAL-version in Europe, NTSC in the US etc.). Maybe changes also have to be made , if you use the new ATARI GTIA chip.

Drawing figures on the screen

I. Drawing a House

Our first elementary example demonstrates how to draw a house on the scrren by using the PLOT and DRAWTO commands. These two commands define screen locations by numbers appended to the commands. For example: PLOT 20,30

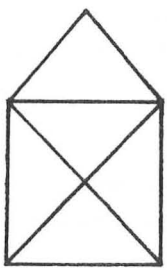
Screen locations:



PLOT x,y

The maximum numbers for x and y depend on the graphics mode that is selected.

Now let' s draw that house.
Most people know how to draw a house with **one** line:



The following program shows how to do that by using the PLOT and DRAWTO commands.

```
10 GRAPHICS 7
20 COLOR 1
30 PLOT 50,60
40 DRAWTO 50,20
45 FOR X=1 TO 200:NEXT X
50 DRAWTO 100,20
55 FOR X=1 TO 200:NEXT X
60 DRAWTO 100,60
65 FOR X=1 TO 200:NEXT X
70 DRAWTO 50,60
75 FOR X=1 TO 200:NEXT X
80 DRAWTO 100,20
85 FOR X=1 TO 200:NEXT X
90 DRAWTO 75,0
95 FOR X=1 TO 200:NEXT X
100 DRAWTO 50,20
105 FOR X=1 TO 200:NEXT X
110 DRAWTO 100,60
```

2. Drawing a Stick Figure

To draw a stick figure or anything else that can be moved around the screen, the PLOT and DRAWTO commands are not very suitable. The ATARI computers have a very powerful feature to do just that. It's called "player-missile" graphics.

This feature offers the following possibilities:

a) we can define four different images (called players) with four different colors
b) with different sizes
c) we can have single or double line resolution
d) each player can have a missile
e) the horizontal position of each player and missile may easily be changed by changing only its position register

Since this technique offers many possibilities, the process of turning on the player-missile graphics is more difficult than drawing something on the screen by the use of PLOT and DRAWTO:

a) first we have to find a location in memory to store our image; this is done by the following command:

I = PEEK (106)–8: POKE 54279,I

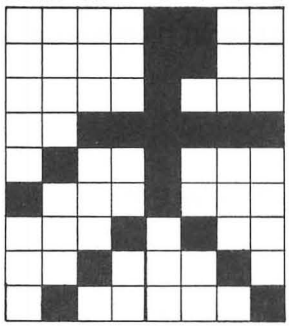
Location 54279 now contains the number of 1/4K blocks from address 0 to the beginning of our player-missile area.
Now we can calculate our player-missile base address by the following command:

PMBAS = I * 256

b) after we have found our player-missile area, we prepare this memory area. What we see on the screen starts 1/2 k after PMBAS and is 128 bytes long. First we want to clear this area, so that there is nothing in it we don' t want. This is done by the following line:

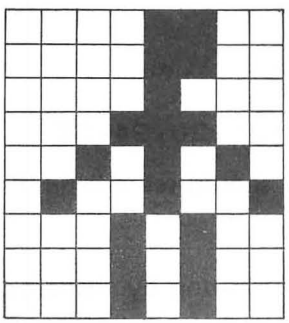
FORX=PMBAS+512 TO PMBAS+640: POKE X,0: NEXT X

Now our PM-area is filled with zeros; that means all dots are turned off. To create a figure we will now turn on some of the dots. Let's create a little man that looks like:



hex		dez.
0C	=	12
0C	=	12
08	=	8
3F	=	63
48	=	72
88	=	136
14	=	20
22	=	34
41	=	65

or if you don't fancy that, try this fellow:



hex		dez.
0C	=	12
0C	=	12
8	=	8
1C	=	28
2A	=	42
49	=	73
14	=	20
14	=	20
14	=	20

f) now we want to position our figure on the screen. The vertical position (y-position) is defined by the position in memory; the horizontal position (x-position) is defined by the contents of the position register for that player which is at location 53248 for player 1.

Let's try:
X = 100 : Y = 70

d) The next step is to define color and luminance of our player. The location for that is 704 (player 1).
Let' s try:
POKE 704,90

e) now we define the resolution of our player and enable the PM-graphics; this is done by:
POKE 559,46 (double line resolution)
POKE 53277,3

But now it is time to see some action on the screen. Let's type in the following program and see what happens on the screen:

```
100 I=PEEK(106)-8
110 POKE 54279,I
120 PMBAS=I*256
130 FOR Q=PMBAS+512 TO PMBAS+640
140 POKE Q,0
150 NEXT Q
160 X=100:Y=70
170 POKE 704,90:POKE 559,46:POKE 53277,3
180 POKE 53248,X
190 FOR Q=0 TO 8
200 READ P
210 POKE PMBAS+512+Y+Q,P
220 NEXT Q
230 DATA 12,12,8,63,72,136,20,34,65
```

When you RUN the program you should see a little man on the screen.
Let's see how we can control him.
First we will change his horizontal position by changing the value in his position register.

POKE 53248,150
should move him to the right.
Now we change his size, which is done by:
POKE 53256,1
or
POKE 53256,3

Then we can change his color by:
POKE 704,60
or
POKE 704,24 (for example)

We can also control the "priority" of our player. That means if he overlays with a part of the playfield, we can define whether our player should be displayed in the foreground or in the background of the playfield.
Let's use the house we made earlier and move our player around that house with different priorities. Type in the following listing and see what happens. (The house is changed a little bit, to show the effect of priority better (see lines 300–330).

```
90 GRAPHICS 7:COLOR 1
100 I=PEEK(106)-8
110 POKE 54279,I
120 PMBAS=I*256
130 FOR Q=PMBAS+512 TO PMBAS+640
140 POKE Q,0
150 NEXT Q
160 X=160:Y=70
170 POKE 704,154:POKE 559,46:POKE 53277,3
180 POKE 53248,X
190 FOR Q=0 TO 8
200 READ P
210 POKE PMBAS+512+Y+Q,P
220 NEXT Q
230 DATA 12,12,8,63,72,136,20,34;65
240 POKE 53256,1
250 PLOT 50,60
260 DRAWTO 50,20:DRAWTO 100,20
270 DRAWTO 100,60:DRAWTO 50,60
280 DRAWTO 100,20:DRAWTO 75,0
290 DRAWTO 50,20:DRAWTO 100,60
```

```
300 FOR Q=1 TO 49
310 PLOT 50+Q,20
320 DRAWTO 50+Q,60
330 NEXT Q
340 POKE 623,8
350 X=X+1:IF X>160 THEN 370
360 POKE 53248,X:GOTO 350
370 POKE 623,0
380 X=X-1:IF X<70 THEN 340
390 POKE 53248,X:GOTO 380
```

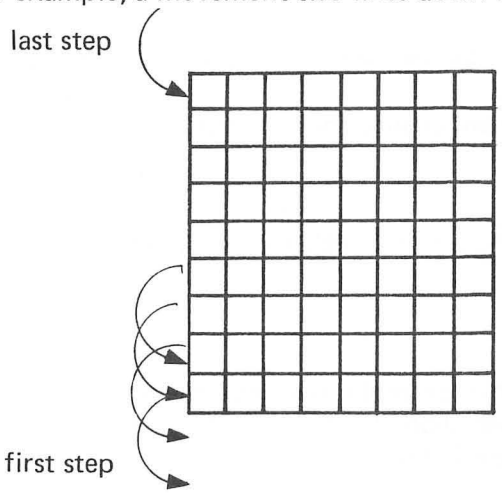
You should see our man moving around the house. Moving from right to left the player has priority over the house (playfield). Moving from left to right the playfield has a higher-priority and the player seems to be behind the house. The change of priority ranking is done in lines 340 and 370. (Memory location 623 = priority control register).

Movements in BASIC

Movements on the screen

In the above examples we could see how easy it is to program horizontal movements of our player by only changing the contents of the position register.
But how about vertical movements?

As mentioned earlier the vertical position of our player is defined by the memory location of the bit pattern of our player. That means that we have to "shovel" our bit pattern to another memory location if we want to move our player in a vertical direction. (Movement up means lower, movement down means higher memory location).
For example, a movement two lines down looks like this:



The following program shows this process of "shoveling". The for-next-loop in lines 245 to 247 takes the bottom line of our player and writes it into the line below. Then the line above is carried one line down and so on. The last step is to take the line above our earlier player (which contains only zeros) and write it on the top line of our earlier player. The movement is now finished.

```
90 GRAPHICS 7:COLOR 1
100 I=PEEK(106)-8
110 POKE 54279,I
120 PMBAS=I*256
130 FOR Q=PMBAS+512 TO PMBAS+640
140 POKE Q,0
150 NEXT Q
160 X=160:Y=70
170 POKE 704,154:POKE 559,46:POKE 53277,3
180 POKE 53248,X
190 FOR Q=0 TO 8
200 READ P
210 POKE PMBAS+512+Y+Q,P
220 NEXT Q
230 DATA 12,12,8,63,72,136,20,34,65
240 POKE 53256,1
245 FOR Q=9 TO 0 STEP -1
246 POKE PMBAS+512+Y+Q,PEEK(PMBAS+511+Y+Q)
247 FOR Z=1 TO 100:NEXT Z:NEXT Q
248 Y=Y+1
249 STOP
```

For a movement upwards we have to begin at the top of our player.
For an upwards movement change the following lines in the above program:

```
245 FOR Q=0 TO 9
246 POKE PMBAS+511+Y+Q,PEEK(PMBAS+512+Y+Q)
247 NEXT Q
248 Y=Y-1
249 GOTO 245
```

As you can see in this example the movement in a vertical direction is much slower than the one in a horizontal direction, even without a delay loop in line 247.

Movements in Machine language

For that reason we will show how to speed up vertical movements by using machine language routines for this job. It is more important here to be able to use these machine language routines than to understand them. The machine language routines do the same as the BASIC statements used earlier except they do it much faster. We will enter the machine language routine in a subroutine. Type in and RUN the following program:

```
100 I=PEEK(106)-8
110 POKE 54279,I
120 PMBAS=I*256
130 FOR Q=PMBAS+512 TO PMBAS+640
140 POKE Q,0
150 NEXT Q
160 X=180:Y=100
170 POKE 704,54:POKE 559,46:POKE 53277,3
180 POKE 53248,X
190 FOR Q=0 TO 8
200 READ P
210 POKE PMBAS+512+Y+Q,P
220 NEXT Q
230 DATA 12,12,8,63,72,136,20,34,65
240 POKE 53256,1
250 GOSUB 1000
260 GOSUB 1100
270 FOR Q=1 TO 80
280 B=USR(UP,PMBAS+511+Y):Y=Y-1
290 NEXT Q
300 FOR Q=1 TO 120
310 X=X-1:POKE 53248,X
320 NEXT Q
330 FOR Q=1 TO 80
340 B=USR(DOWN,PMBAS+511+Y):Y=Y+1
350 NEXT Q
```

A similar program was published in COMPUTE 4/81.

```
360 FOR Q=1 TO 120
370 X=X+1:POKE 53248,X
380 NEXT Q
390 GOTO 270
1000 DIM UPCODE$(21):UP=ADR(UPCODE$)
1010 FOR I=UP TO UP+20
1020 READ B:POKE I,B
1030 NEXT I:RETURN
1040 DATA 104,104,133,204,104,133,203
1050 DATA 160,1,177,203,136,145,203
1060 DATA 200,200,192,11,208,245,96
1100 DIM DOWNCODE$(21):DOWN=ADR(DOWNCODE$)
1110 FOR I=DOWN TO DOWN+20
1120 READ B:POKE I,B
1130 NEXT I:RETURN
1140 DATA 104,104,133,204,104,133,203
1150 DATA 160,10,177,203,200,145,203
1160 DATA 136,136,192,255,208,245,96
```

This program moves our player up and left and down and right through the listing of the program. What' s new here is the fact that vertical and horizontal movements happen with about the same speed.

The subroutine in lines 1000—1060 writes a machine language routine for upward movements into memory which can be executed directly by the microprocessor. The instructions are stored in the DATA-statements in lines 1040, 1050, 1060. The subroutine in lines 1100—1160 does the same for downward movements.

Those subroutines are called in lines 280 (upward movement) and line 340 (downward movement).

You can use these machine language routines for any image that's 9 lines high and also if you use double line resolution (which is selected in line 170 in this example by: POKE 559,46).

Alternating Shapes

So far we have learned how to draw a player and move it around the screen. But the player seems to have no movement in itself. In this chapter we will try to change the shape of the player when it moves. This can be done by using two different shapes and, for example, writing them alternately into the player-missile area.

```
100 DIM P$(54),V$(27),D$(3)
110 FOR I=1 TO 18
120 READ D$
130 P$(3*I-2,3*I)=D$
140 NEXT I
150 GRAPHICS 18:SETCOLOR 4,7,2
160 X=50:Y=60
170 POKE 559,46
180 I=PEEK(106)-8:POKE 54279,I
190 PB=256*I
200 POKE 53277,3
210 FOR I=PB+512 TO PB+640
220 POKE 1,0
230 NEXT I
240 POKE 704,22
250 POKE 53248,X
255 POKE 53256,1
260 C=C+1
270 IF C>3 THEN C=1
280 ON C GOTO 290,300,290
290 V$=P$(1,27):GOTO 310
300 V$=P$(28,54)
310 FOR I=1 TO 9
320 POKE PB+512+Y+I,VAL(V$(3*I-2,3*I))
330 NEXT I
340 X=X+2:POKE 53248,X
350 IF X>190 THEN 350
360 GOTO 260
400 DATA 012,012,008,063,072,136,020,034,065
410 DATA 012,012,008,028,042,073,020,020,020
```

If you RUN this program you should see that there are two alternating shapes written into the player-missile area. In addition, the position register is changed so it looks like our player is running from left to right across the screen.

Let's have a closer look at that program.
In lines 110—140 the data for both players are read into a string (P$). Lines 260—300 select alternately the lower and upper substring of P$. In lines 310—330 this substring is converted back to a number and written into the player-missile area. Line 340 changes the position register.

Movements of Missiles

Now that we know how to create and move players on the screen, it is time to use the missile as well. The next example shows the use of player 0 and missile 0. Player and missile use the same color register but different position registers. Consequently, they can be moved independently. The location in memory for the missiles is PMBAS+384 (if you use double line resolution). This is for all four missiles together which means that there are only two bits in a row per missile which can be turned on or off. We turn on only one bit which is done in line 335 of the following listing. You will note that the location is not PB+384+Y but four more; the reason is that we want the bullet to come out of the hand of our little man and not out of his head.

```
100 DIM P$(27),D$(3)
110 FOR I=1 TO 9
120 READ D$
130 P$(3*I-2,3*I)=D$
140 NEXT I
150 GRAPHICS 18:SETCOLOR 4,7,2
160 X=50:Y=60
170 POKE 559,46
180 I=PEEK(106)-8:POKE 54279,I
190 PB=256*I
200 POKE 53277,3
210 FOR I=PB+512 TO PB+640
220 POKE I,0
230 NEXT I
240 POKE 704,22
250 POKE 53248,X
260 POKE 53256,1
310 FOR I=1 TO 9
320 POKE PB+512+Y+I,VAL(P$(3*I-2,3*I))
330 NEXT I
335 POKE PB+388+Y,1
```

```
340 X=X+1:POKE 53252,X
350 IF X>190 THEN X=55
360 GOTO 340
400 DATA 012,012,008,063,072,136,020,034,065
```

The next step is to expand the example to two players and two missiles. Try the following:

```
100 I=PEEK(106)-8:POKE 54279,I
110 PB=256*I
120 X1=50:Y1=60:X2=180:Y2=60
130 FOR Q=PB+384 TO PB+768
140 POKE Q,0
150 NEXT Q
160 GRAPHICS 18:SETCOLOR 4,7,2
170 POKE 559,46
180 POKE 53277,3
190 POKE 704,22:POKE 705,90
200 POKE 53248,X1:POKE 53249,X2
210 POKE 53256,1:POKE 53257,1
220 FOR Q=1 TO 9
230 READ P
240 POKE PB+512+Y1+Q,P
250 NEXT Q
260 FOR Q=1 TO 9
270 READ P
280 POKE PB+640+Y2+Q,P
290 NEXT Q
300 POKE PB+384+4+Y1,5
310 FOR Q=1 TO 150
320 POKE 53252,X1+Q:POKE 53253,X2-Q
330 NEXT Q
340 GOTO 310
400 DATA 12,12,8,63,72,136,20,34,65
410 DATA 12,12,8,28,42,73,20,20,20
```

This program shows two players shooting at each other. If you watch that program you probably will miss the determination whether a player is hit by a missile.

Overlapping Detection (Collision Detection)

To determine overlappings between missiles, players and playfield, the ATARI computer has 16 registers that indicate collisions. The register that indicates collisions between missile 0 and player is at address 53256 (for details about hardware registers see the appendix).

Once a collision has occured, the indication remains in that collision register until we write something into a register that is called "collision clear " (address 53278).
Change the following lines in the program and watch the display of the contents of the collision register.

```
160 GRAPHICS 8:SETCOLOR 4,7,2
320 POKE 53252,X1+Q
330 ? PEEK(53256):POKE 53278,0:NEXT Q
```

You should see a "0" while missile 0 is between the two players, a "2" when player 1 is hit and a "1" when missile 0 comes out of player 0 (left player).
Now that we know when the other player has been hit, we can create some kind of an "explosion routine". That means we have to change the shape of the player that was hit. The following sample shows how this can be done.

```
100 I=PEEK(106)-8:POKE 54279,I
110 PB=256*I
120 X1=50:Y1=60:X2=180:Y2=60
130 FOR Q=PB+384 TO PB+768
140 POKE Q,0
150 NEXT Q
160 GRAPHICS 18:SETCOLOR 4,7,2
170 POKE 559,46
180 POKE 53277,3
```

```
190 POKE 704,22:POKE 705,90
200 POKE 53248,X1:POKE 53249,X2
210 POKE 53256,1:POKE 53257,1
220 FOR Q=1 TO 9
230 READ P
240 POKE PB+512+Y1+Q,P
250 NEXT Q
260 FOR Q=1 TO 9
270 READ P
280 POKE PB+640+Y2+Q,P
290 NEXT Q
300 POKE PB+384+4+Y1,5
310 FOR Q=1 TO 150
320 POKE 53252,X1+Q
330 IF PEEK(53256)=2 THEN GOSUB 500:RESTORE
    :GOTO 220
335 POKE 53278,0
340 NEXT Q
350 RESTORE :GOTO 220
400 DATA 12,12,8,63,72,136,20,34,65
410 DATA 12,12,8,28,42,73,20,20,20
500 FOR Q=20 TO 9 STEP -1:POKE 705,10*Q:NEXT Q
505 POKE 53252,1
510 FOR Q=1 TO 9
520 POKE PB+640+Y2+Q,0
530 NEXT Q
540 POKE PB+640+Y2+8,24
550 POKE PB+640+Y2+9,255
560 POKE 53278,0
570 FOR Q=1 TO 1000:NEXT Q
580 RETURN
```

This program is similar to the one before. Line 330 checks the collision register. If a collision is detected, the program branches to the subroutine in lines 500–580. Line 335 clears the collision registers. Line 350 is necessary in case the collision is not detected which happens sometimes because of the slowness of BASIC. Line 500 changes the color of the player very quickly which makes him shake.

Line 505 puts the bullet back. Lines 510–530 erase the player. Lines 540 and 550 create a new player (prone). Line 560 clears the collision registers. Line 570 is a delay loop.

Sound features

In the chapters we have discussed so far you probably missed one feature of the ATARI computers and that is SOUND.

The ATARI computers have four independent channels to generate sound and noise. For each channel you can define the frequency, the kind of sound (noise or tone) and the volume.
The command looks like this:

SOUND V, N, T, L

where V is voice, N is note, T is tone and L is loudness.
The kind of sound most often needed in games is a shot.
If you type in: SOUND 0, 5, 0, 8
you will hear a noise that may be used for a flying jet.
If we manipulate the volume of that noise we can create a shot, a steam engine or a noise similar to the one at the beach. It is easy to change shot to an explosion, just change the delay in line 40.

```
10 REM SHOT
20 FOR X=15 TO 0 STEP -1
30 SOUND 0,5,0,X
40 FOR Q=1 TO 3:NEXT Q
50 NEXT X
60 FOR Q=1 TO 500:NEXT Q
70 GOTO 10
```

```
10 REM STEAM LOCOMOTIVE
20 FOR X=15 TO 0 STEP -1
30 SOUND 0,2,0,X
40 NEXT X
50 FOR Q=1 TO 30:NEXT Q
60 GOTO 10
```

```
10 REM OCEAN
20 FOR X=0 TO 15
30 SOUND 0,2,0,X
40 FOR Q=1 TO 70:NEXT Q
50 NEXT X
60 FOR X=15 TO 0 STEP -1
70 SOUND 0,2,0,X
80 FOR Q=1 TO 50:NEXT Q
90 NEXT X
100 GOTO 20
```

To discover the sound you need for your own game you can use the following program:

```
10 REM SOUNDGENERATOR
20 FOR N=0 TO 255 STEP 5
30 FOR T=0 TO 7
40 FOR L=0 TO 15
50 PRINT "N=";N;"  ";"T=";T;"  ";"L=";L
60 SOUND 0,N,T,L
70 NEXT L
80 NEXT T
90 NEXT N
```

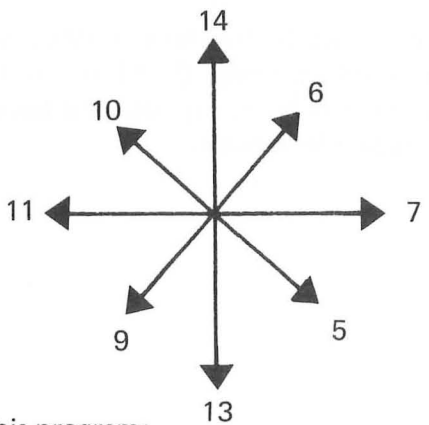
If you like a sound generated by the above program press BREAK. The sound will stay until you enter CONT to continue with the program. To create more complex sounds you have to use two, three or four channels simultaneously.

Programming the Joystick

Another feature of the ATARI computers are the four joystick connectors. Connect a joystick to the left connector and run the following program:

```
10 PRINT STICK(0)
20 GOTO 10
```

You will get the following values, depending on the position of your joystick:



If you run this program:

```
10 PRINT STRIG(0)
20 GOTO 10
```

you will get a "1" if the joystick is not being used and a "0" if you press the red trigger button on your joystick.

Backgammon

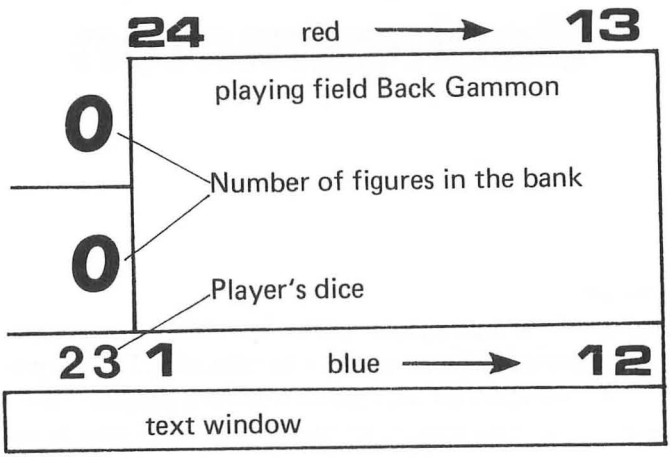
Backgammon

Backgammon is a two-player game in which opponents try to bring their figures home as quickly as possible. The winner is the one who first removes all his figures from the playing field. On the lower left corner the computer shows the dice points he threw.

A player must move any of his figures the same number of "points" as is shown on either of the dice. Then he must again move a figure (which may be the same figure again) the same number as the second die. When a player moves, he may not finally land on a point occupied by two or more of his opponent's figures.

If there is a "double" (both dice are the same) you may move four times.
If a figure is left alone on any "point" it is a "blot". The opponent may "hit" the "blot" by moving his figure onto this field if his dice roll permits. The figure which has been "hit" is moved back on to the "bar". The figures on the "bar" re-enter the playing field again before any other of that player's figures is moved. To re-enter, a player must roll a number 1 thru 6 not occupied by two or more of his opponents figures. If you want to bring a figure "home", all your other figures must be on the final six fields. If you can't move, then hit only "0" after the computer's question "from".

Load the program and start it with "RUN". The computer displays the name of the game. To start, press RETURN. The playing field will be plotted to the screen and the computer asks who wants to begin. (red = 1 — blue = 2). Enter the number with RETURN. The computer will throw the dice and display the points in the color of the player who has the next move.



Red moves from the higher to the lower numbers, blue vice versa. Now enter from where and to where you choose to move and then enter RETURN. If a move is impossible, the computer will print "sorry, no move is possible" and wait for RETURN.
If some figures are on the bar, the computer will print "from bar to" and wait for the input.
To bring any figure home, red must enter "to 0" and blue " to 25".

```
10 DIM A(26),B(26),A$(1)
20 GRAPHICS 2:? #6:? #6;"  BaCkGaMmon"
30 POKE 710,152:INPUT A$
1000 GRAPHICS 7:POKE 752,1
1010 SETCOLOR 0,3,10:SETCOLOR 1,8,12:SETCOL
OR 2,12,3
1060 COLOR 3:PLOT 20,70:DRAWTO 20,10:DRAWTO
 159,10:DRAWTO 159,70:DRAWTO 20,70
1070 PLOT 21,69:DRAWTO 21,11:DRAWTO 158,11:
DRAWTO 158,69:DRAWTO 21,69
1072 PLOT 0,40:DRAWTO 20,40:PLOT 0,39:DRAWT
O 20,39
1074 PLOT 0,69:DRAWTO 20,69:DRAWTO 20,79:PL
OT 0,70:DRAWTO 21,70:DRAWTO 21,79
1080 C=0:Z=10:GOSUB 4000
1090 C=1:Z=45:GOSUB 4000
1140 Z=0:X=37:COLOR 2:GOSUB 4120:X=43:GOSUB
 4130
1141 X=59:GOSUB 4120:X=65:GOSUB 4110
```

```
1142 X=81:GOSUB 4110:X=87:GOSUB 4190
1143 X=103:GOSUB 4110:X=109:GOSUB 4170
1144 X=125:GOSUB 4110:X=131:GOSUB 4150
1145 X=147:GOSUB 4110:X=153:GOSUB 4130
1150 X=26:COLOR 1:GOSUB 4120:X=32:GOSUB 414
0
1160 X=48:GOSUB 4120:X=54:GOSUB 4120
1170 X=70:GOSUB 4120:X=76:GOSUB 4100
1190 X=92:GOSUB 4110:X=98:GOSUB 4180
1210 X=114:GOSUB 4110:X=120:GOSUB 4160
1230 X=136:GOSUB 4110:X=142:GOSUB 4140
1270 Z=73:X=38:GOSUB 4120
1271 X=60:GOSUB 4140:X=82:GOSUB 4160:X=104:
GOSUB 4180
1274 X=124:GOSUB 4110:X=130:GOSUB 4100
1275 X=147:GOSUB 4110:X=153:GOSUB 4120
1279 COLOR 2:X=27:GOSUB 4110:X=49:GOSUB 413
0:X=71:GOSUB 4150:X=93:GOSUB 4170:X=115:GOS
UB 4190
1360 X=135:GOSUB 4110:X=141:GOSUB 4110
1380 FOR M=0 TO 26:A(M)=0:B(M)=0:NEXT M
1390 B(1)=2:B(12)=5:B(17)=3:B(19)=5:A(24)=2
:A(13)=5:A(8)=3:A(6)=5
1400 X=30:Z=13:COLOR 1:GOSUB 4200
1410 Z=17:GOSUB 4200
1420 X=151:FOR Z=13 TO 29 STEP 4
1425 GOSUB 4200:NEXT Z
1430 X=85:FOR Z=50 TO 66 STEP 4
1435 GOSUB 4200:NEXT Z
1440 X=107:FOR Z=58 TO 66 STEP 4
1445 GOSUB 4200:NEXT Z
1450 COLOR 2
1460 X=151:FOR Z=50 TO 66 STEP 4
1465 GOSUB 4200:NEXT Z
1470 X=85:FOR Z=13 TO 29 STEP 4
1475 GOSUB 4200:NEXT Z
1480 X=107:FOR Z=13 TO 21 STEP 4
1485 GOSUB 4200:NEXT Z
1490 X=30:FOR Z=62 TO 66 STEP 4
1495 GOSUB 4200:NEXT Z
1500 X=10:Z=20:GOSUB 4100
1505 Z=53:COLOR 1:GOSUB 4100
1510 TRAP 1510:? "Player 'RED' =1: Player '
```

```
BLUE' =2":INPUT W:IF W=2 THEN 2500
1550 A0=0:B25=0
1600 ? "}":N1=0:G=4:S=1:GOSUB 5000
1605 GOSUB 2000
1610 IF C1=C2 THEN 1700
1615 G=2:E=0
1690 A0=A0+A(0):A(0)=0
1700 GOSUB 6000
1710 E=E1+E2:IF E>0 THEN 1750
1720 ? "I'm sorry but you can't move!":INPU
T A$:GOTO 2500
1750 GOSUB 3000
1755 IF N1=1 THEN 2500
1760 GOSUB 3200
1770 IF B1=1 THEN 1750
1772 GOSUB 3400:GOSUB 10000
1775 G=G-1:IF G=0 THEN 2500
1780 GOTO 1690
2000 IF A(0)+A0=15 THEN 11000
2010 IF B25+B(25)=15 THEN 11100
2020 RETURN
2500 ? "}":N1=0:G=4:S=2:GOSUB 5000
2505 GOSUB 2000
2510 IF C1=C2 THEN 2600
2515 G=2:E=0
2590 B25=B(25)+B25:B(25)=0
2600 GOSUB 6100
2610 E=E1+E2:IF E>0 THEN 2650
2620 ? "I'm sorry but you can't move!":INPU
T A$:GOTO 1600
2647 GOTO 1600
2650 GOSUB 3000
2655 IF N1=1 THEN 1600
2660 GOSUB 9200
2670 IF B1=1 THEN 2650
2672 GOSUB 9400:GOSUB 10000
2675 G=G-1:IF G=0 THEN 1600
2680 GOTO 2590
3000 IF S=2 THEN 9000
3001 IF AA>0 THEN 3090
3003 TRAP 3003:? "from ",:INPUT A1:IF A1=0
THEN 12000
3010 TRAP 3010:? "to ",:INPUT A2:F1=0:F2=0
```

```
3017 IF S=2 THEN F=A2-A1
3020 IF S=1 THEN F=A1-A2
3030 IF F=C1 THEN 3060
3040 IF F=C2 THEN 3080
3050 ? "Sorry, but you can't move this way"
:GOTO 3000
3060 F1=1:RETURN
3080 F2=1:RETURN
3090 A1=25:? "From Bank ";:GOTO 3010
3200 IF AA>0 THEN 3203
3201 IF A(A1)=0 THEN 3215
3203 IF A2=0 THEN 3270
3205 IF B(A2)=0 THEN 3230
3210 IF B(A2)=1 THEN 3220
3215 ? "Sorry, but you can't move this way"
:B1=1:RETURN
3220 B(A2)=0:BB=BB+1:X=10:Z=20:GOSUB 4300
3225 COLOR 2:ON BB GOSUB 4110,4120,4130,414
0,4150,4160,4170,4180,4190
3230 IF A1=25 THEN 3245
3232 A(A1)=A(A1)-1:A(A2)=A(A2)+1
3240 B1=0:RETURN
3245 X=10:Z=53:COLOR S:GOSUB 4300
3247 AA=AA-1:A(A2)=A(A2)+1
3250 COLOR 1:ON AA+1 GOSUB 4100,4110,4120,4
130,4140,4150,4160,4170,4180
3260 GOTO 3240
3270 IF L=0 THEN 3205
3275 GOTO 3215
3400 IF A(A1)>5 THEN 3450
3402 IF A1=25 THEN 3450
3405 A=A1:GOSUB 8000
3410 A=A(A1)+1:IF A1>12 THEN GOTO 3480
3415 A=A(A1)+1:IF A1<13 THEN GOSUB 8300
3420 COLOR 0:GOSUB 4200
3450 IF A(A2)>6 THEN 3475
3452 IF A2=0 THEN RETURN
3455 A=A2:GOSUB 8000
3460 A=A(A2):IF A2>12 THEN GOTO 3490
3465 A=A(A2):IF A2<13 THEN GOSUB 8300
3470 COLOR 1:GOSUB 4200:RETURN
3480 GOSUB 8200:GOTO 3420
3490 GOSUB 8200:GOTO 3470
```

```
4000 FOR X=30 TO 151 STEP 11:C=C+1:IF C=3 T
HEN C=1
4010 COLOR C:PLOT X,Z:DRAWTO X,Z+25:PLOT X-
1,Z:DRAWTO X-1,Z+25:NEXT X:RETURN
4100 PLOT X,Z:DRAWTO X+3,Z:DRAWTO X+3,Z+6:D
RAWTO X,Z+6:DRAWTO X,Z:RETURN
4110 PLOT X+2,Z:DRAWTO X+2,Z+6:RETURN
4120 PLOT X,Z:DRAWTO X+3,Z:DRAWTO X+3,Z+3:D
RAWTO X,Z+3:DRAWTO X,Z+6:DRAWTO X+3,Z+6:RET
URN
4130 PLOT X,Z:DRAWTO X+3,Z:DRAWTO X+3,Z+6:D
RAWTO X,Z+6:PLOT X,Z+3:DRAWTO X+3,Z+3:RETUR
N
4140 PLOT X+3,Z:DRAWTO X+3,Z+6:PLOT X,Z:DRA
WTO X,Z+3:DRAWTO X+3,Z+3:RETURN
4150 PLOT X+3,Z:DRAWTO X,Z:DRAWTO X,Z+3:DRA
WTO X+3,Z+3:DRAWTO X+3,Z+6:DRAWTO X,Z+6:RET
URN
4160 PLOT X+3,Z:DRAWTO X,Z:DRAWTO X,Z+6:DRA
WTO X+3,Z+6:DRAWTO X+3,Z+3:DRAWTO X,Z+3:RET
URN
4170 PLOT X,Z:DRAWTO X+2,Z:DRAWTO X+2,Z+6:R
ETURN
4180 PLOT X,Z:DRAWTO X+3,Z:DRAWTO X+3,Z+6:D
RAWTO X,Z+6:DRAWTO X,Z:PLOT X+3,Z+3:DRAWTO
X,Z+3:RETURN
4190 PLOT X,Z+6:DRAWTO X+3,Z+6:DRAWTO X+3,Z
:DRAWTO X,Z:DRAWTO X,Z+3:DRAWTO X+3,Z+3:RET
URN
4200 FOR M=Z TO Z+1:PLOT X-4,M:DRAWTO X-2,M
:PLOT X+1,M:DRAWTO X+3,M:NEXT M:RETURN
4300 COLOR 0:FOR A=X TO X+4:PLOT A,Z:DRAWTO
 A,Z+6:NEXT A:RETURN
4310 COLOR 0:FOR Z=71 TO 79:PLOT 0,Z:DRAWTO
 19,Z:NEXT Z:RETURN
4500 C=INT(RND(1)*6)+1:RETURN
5000 GOSUB 4310
5005 GOSUB 4500:C1=C
5010 GOSUB 4500:C2=C
5020 X=1:Z=71:COLOR S
5030 ON C1 GOSUB 4110,4120,4130,4140,4150,4
160
5040 X=11:ON C2 GOSUB 4110,4120,4130,4140,4
150,4160
```

```
5050 RETURN
6000 L=1:N=A0+A(1)+A(2)+A(3)+A(4)+A(5)+A(6)
:IF N=15 THEN L=0
6005 E1=0:E2=0:IF AA>0 THEN 6999
6010 FOR D=1 TO 24
6020 IF A(D)>0 THEN GOSUB 7000
6030 NEXT D:RETURN
6100 L=24:N=B25+B(24)+B(23)+B(22)+B(21)+B(2
0)+B(19):IF N=15 THEN L=25
6105 E1=0:E2=0:IF BB>0 THEN 7099
6110 FOR D=1 TO 24
6120 IF B(D)>0 THEN GOSUB 7100
6130 NEXT D:RETURN
6999 D=25
7000 D1=D-C1:IF (D1)<L THEN 7030
7010 IF B(D1)>1 THEN 7030
7020 E1=1
7030 D1=D-C2:IF (D1)<L THEN RETURN
7040 IF B(D1)>1 THEN RETURN
7050 E2=1:RETURN
7099 D=0
7100 D1=D+C1:IF (D1)>L THEN 7130
7105 IF D1>25 THEN 7130
7110 IF A(D1)>1 THEN 7130
7120 E1=1
7130 D1=D+C2:IF (D1)>L THEN RETURN
7135 IF D1>25 THEN RETURN
7140 IF A(D1)>1 THEN RETURN
7150 E2=1:RETURN
8000 IF A>20 THEN 8005
8001 ON A GOTO 8010,8020,8030,8040,8050,806
0,8070,8080,8090,8100,8110,8120,8120,8110,8
100,8090,8080,8070,8060,8050
8005 A=A-20:ON A GOTO 8040,8030,8020,8010
8010 X=30:RETURN
8020 X=41:RETURN
8030 X=52:RETURN
8040 X=63:RETURN
8050 X=74:RETURN
8060 X=85:RETURN
8070 X=96:RETURN
8080 X=107:RETURN
8090 X=118:RETURN
```

```
8100 X=129:RETURN
8110 X=140:RETURN
8120 X=151:RETURN
8200 ON A GOTO 8210,8220,8230,8240,8250,826
0
8210 Z=13:RETURN
8220 Z=17:RETURN
8230 Z=21:RETURN
8240 Z=25:RETURN
8250 Z=29:RETURN
8260 Z=33:RETURN
8300 ON A GOTO 8310,8320,8330,8340,8350,836
0
8310 Z=66:RETURN
8320 Z=62:RETURN
8330 Z=58:RETURN
8340 Z=54:RETURN
8350 Z=50:RETURN
8360 Z=46:RETURN
9000 IF BB<1 THEN 3003
9010 A1=0:? " From Bank ";:GOTO 3010
9200 IF BB>0 THEN 9203
9201 IF B(A1)=0 THEN 9215
9203 IF A2=25 THEN 9270
9205 IF A(A2)=0 THEN 9230
9210 IF A(A2)=1 THEN 9220
9215 ? "Sorry, but you can't move this way"
:B1=1:RETURN
9220 A(A2)=0:AA=AA+1:X=10:Z=53:GOSUB 4300
9225 COLOR 1:ON AA GOSUB 4110,4120,4130,414
04150,4160,4170,4180,4190
9230 IF A1=0 THEN 9245
9232 B(A1)=B(A1)-1:B(A2)=B(A2)+1
9240 B1=0:RETURN
9245 X=10:Z=20:COLOR 5:GOSUB 4300
9247 BB=BB-1:B(A2)=B(A2)+1
9250 COLOR 2:ON BB+1 GOSUB 4100,4110,4120,4
130,4140,4150,4160,4170,4180,4170,4180
9260 GOTO 9240
9270 IF L=25 THEN 9205
9275 GOTO 9215
9400 IF B(A1)>5 THEN 9450
9402 IF A1=0 THEN 9450
```

```
9405 A=A1:GOSUB 8000
9410 A=B(A1)+1:IF A1>12 THEN GOTO 9480
9415 A=B(A1)+1:IF A1<13 THEN GOSUB 8300
9420 COLOR 0:GOSUB 4200
9450 IF B(A2)>6 THEN 9475
9452 IF A2=25 THEN RETURN
9455 A=A2:GOSUB 8000
9460 A=B(A2):IF A2>12 THEN GOTO 9490
9465 A=B(A2):IF A2<13 THEN GOSUB 8300
9470 COLOR 2:GOSUB 4200
9475 RETURN
9480 GOSUB 8200:GOTO 9420
9490 GOSUB 8200:GOTO 9470
10000 IF C1=C2 THEN RETURN
10010 IF F=C1 THEN C1=50
10020 IF F=C2 THEN C2=50
10030 RETURN
11000 ? "RED is the winner":END
11100 ? "Blue is the winner":END
12000 IF N=15 THEN 12100
12010 ? "You have to move!":GOTO 3003
12100 N1=1:RETURN
```

SMARTY

SMARTY

Load the program and start it with RUN. The computer displays the name of the game and asks you whether or not you need instructions. Enter "Y" or "N" and Return. After reading the instructions, press START to begin the game. In case of "N" the computer will begin immediately.

First the computer will ask how many positions you want to guess (3–10). Enter the number and the RETURN.

After that the computer will ask how many characters you want (5 – 16). Enter the number and the RETURN. If you choose "5" then the possible characters are 0, 1, 2, 3, 4, 5; if you choose "16" then the possible characters are 0, 1, 2...9, A, B, ...G. Each time the combination is N+1 characters (N being number you entered).

Now the computer looks for a combination and you guess it. The characters are displayed on the left side of the graphics window. A wrong character cannot be deleted!!

The computer checks for correct characters. For each character at the correct position the computer displays a green "X". For every other correct character at a wrong position a violet "X".

```
10 DIM A(21),B(21),D$(300),C(10),CC(10),D(2
1)
20 DIM A$(10),B$(10),C$(1):AB=0:A(0)=0
30 GRAPHICS 2:POKE 85,5:? #6;"   SMARTY   "
42 FOR A=1 TO 20:A(A)=0:B(A)=0:D(A)=10:NEXT
 A:FOR A=1 TO 300:D$(A,A)="-":NEXT A
43 IF A(0)=0 THEN GOSUB 3000
45 A(0)=0:POKE 84,8:? #6;"!3-10! positions
are possible"
50 TRAP 50:INPUT A
52 IF A<3 OR A>10 THEN 45
55 POKE 84,8:? #6;"!5-10! characters   are
possible"
60 TRAP 60:INPUT B
```

```
62 IF B<5 OR B>16 THEN 55
200 POKE 84,8:? #6;"I look for a GOOD   Com
BInaTIon "
210 FOR C=1 TO A:D=INT(RND(0)*(B+1)):GOSUB
220:A$(C,C)=C$:NEXT C:GOTO 290
220 IF D=10 THEN C$="A"
221 IF D=11 THEN C$="B"
222 IF D=12 THEN C$="C"
223 IF D=13 THEN C$="D"
224 IF D=14 THEN C$="E"
225 IF D=15 THEN C$="F"
226 IF D=16 THEN C$="G"
227 IF D=0 THEN C$="0"
231 IF D=1 THEN C$="1"
232 IF D=2 THEN C$="2"
233 IF D=3 THEN C$="3"
234 IF D=4 THEN C$="4"
235 IF D=5 THEN C$="5"
236 IF D=6 THEN C$="6"
237 IF D=7 THEN C$="7"
238 IF D=8 THEN C$="8"
239 IF D=9 THEN C$="9"
240 RETURN
290 FOR C=1 TO 999:NEXT C
295 GRAPHICS 1:SETCOLOR 2,2,2:SETCOLOR 0,2,
10:POKE 752,1:POKE 54279,56:POKE 559,46:GOS
UB 2000
300 GOSUB 4100:D=0:AB=AB+1:FOR C=1 TO 181 S
TEP 10:D=D+1:POKE 84,D-1:POKE 85,10-A
305 FOR CC=C TO C+A-1:D$(CC,CC)=D$(CC+10,CC
+10):NEXT CC
306 ? #6;D$(C,C+A-1);:GOSUB 310:NEXT C:GOTO
 350
310 A(D)=A(D+1):IF A(D)=0 THEN 320
315 FOR E=1 TO A(D):? #6;"x";:NEXT E
320 B(D)=B(D+1):IF B(D)=0 THEN 330
325 FOR E=1 TO B(D):? #6;"x";:NEXT E
330 D(D)=D(D+1):IF D(D)=0 THEN 340
335 FOR E=1 TO D(D):? #6;" ";:NEXT E
340 RETURN
350 POKE 84,19:POKE 85,10-A
355 FOR C=1 TO 2*A:? #6;" ";:NEXT C
358 POKE 84,19:POKE 85,10-A:GOSUB 4000
```

```
359 ? "Your Combination: ":? "pass ";AB
360 FOR C=1 TO A:GOSUB 1000:? #6;C$;:B$(C,C
)=C$:NEXT C
370 POKE 764,255:? "}"
380 FOR C=1 TO 10:C(C)=0:CC(C)=0:NEXT C:A(2
0)=0:B(20)=0
390 FOR C=1 TO A:IF A$(C,C)=B$(C,C) THEN C(
C)=1:A(20)=A(20)+1
400 NEXT C:FOR C=1 TO A
405 IF C(C)=1 THEN 430
410 FOR CC=1 TO A:IF CC(CC)=1 THEN 420
412 IF C(CC)=1 THEN 420
415 IF A$(C,C)=B$(CC,CC) THEN B(20)=B(20)+1
:CC(CC)=1:GOTO 430
420 NEXT CC
430 NEXT C
440 FOR C=1 TO A:D$(190+C)=B$(C):NEXT C
441 D(20)=10-A(20)-B(20)
442 D(21)=D(20):A(21)=A(20):B(21)=B(20)
445 D=20:GOSUB 310
450 IF A(20)=A THEN 500
460 GOTO 300
500 ? :? :? :? "You found the combination":
? "You tried ";AB;" times":? "Press <START>
 for another game"
510 ? "<SELECT> to end";
520 IF PEEK(53279)=6 THEN POKE 53277,4:POKE
 53248,0:RUN
530 IF PEEK(53279)=5 THEN POKE 53277,4:POK
E 53248,0:END
540 GOTO 520
999 END
1000 POKE 764,255
1010 IF PEEK(764)=50 THEN C$="0":RETURN
1011 IF PEEK(764)=31 THEN C$="1":RETURN
1012 IF PEEK(764)=30 THEN C$="2":RETURN
1013 IF PEEK(764)=26 THEN C$="3":RETURN
1014 IF PEEK(764)=24 THEN C$="4":RETURN
1015 IF PEEK(764)=29 THEN C$="5":RETURN
1016 IF PEEK(764)=27 THEN C$="6":RETURN
1017 IF PEEK(764)=51 THEN C$="7":RETURN
1018 IF PEEK(764)=53 THEN C$="8":RETURN
1019 IF PEEK(764)=48 THEN C$="9":RETURN
```

```
1020 IF PEEK(764)=63 THEN C$="A":RETURN
1021 IF PEEK(764)=21 THEN C$="B":RETURN
1022 IF PEEK(764)=18 THEN C$="C":RETURN
1023 IF PEEK(764)=58 THEN C$="D":RETURN
1024 IF PEEK(764)=42 THEN C$="E":RETURN
1025 IF PEEK(764)=56 THEN C$="F":RETURN
1026 IF PEEK(764)=61 THEN C$="G":RETURN
1030 GOTO 1010
2000 FOR E=14860 TO 14943
2010 POKE E,1:NEXT E
2020 FOR E=14852 TO 14859:POKE E,0:NEXT E
2030 FOR E=14944 TO 14972:POKE E,0:NEXT E
2090 POKE 704,222:POKE 53256,2:POKE 53248,1
21:POKE 53277,2:POKE 54279,56
2100 RETURN
3000 POKE 84,8:? #6;"DO YOU NEED INSTRUCTIO
NS":? " ! Y-N !"
3010 INPUT C$:IF C$="Y" THEN 3030
3020 ? "}":POKE 84,8:? #6;"
                          ":RETURN
3030 GRAPHICS 0:SETCOLOR 2,0,0:SETCOLOR 4,0
,0:SETCOLOR 1,3,8
3031 POKE 752,1:J=6+PEEK(742)*256+PEEK(741)

3032 A(0)=1:POKE J+1,7:POKE 82,2:POKE 752,0
:POKE J+2,6
3035 ? :? " MASTER mind    ":? :? "I will l
ook for a combination,":? "and you may gues
s it!"
3040 ? "The combination consists of the":?
"numbers 0-9 and the letters ABCDEFG."
3050 ? "You select the difficulty by choosi
ng":? "the numbers of positions and":? "cha
racters"
3070 ? "For any character at the correct":?
 "position I will set a green 'X'."
3075 ? "For any correct character at a wron
g"
3080 ? "position I will set a violet 'X'."
3100 ? :? "Good luck!"
3110 ? "Press <START> to begin."
3120 IF PEEK(53279)=6 THEN 30
3130 GOTO 3120
```

```
4000 POKE 54272,QC:POKE 53774,NQ:POKE 54286
,MI:POKE 53248,121:RETURN
4100 MI=PEEK(54286):NQ=PEEK(53774):QC=PEEK(
54272):POKE 54286,0:POKE 53774,0:POKE 54272
,0:POKE 53248,10:RETURN
```

BOMBER

BOMBER

Bomber is a game that demonstrates how to use the P/M graphics of your ATARI computer and how to simulate the flight of a bomb or projectile.

The program uses players 0 to 3 and missiles 0 and 1. Player 0 is the bomber; player 1 the gun. The two other players are used to simulate the explosions of players 0 and 1. Missile 0 is the aircraft bomb; missile 1 the gun projectile. The players are defined the same way each time. First calculate the P/M base address and store it (line 40 and 50). Then set P/M address (line 60). Then decide on single line resolution (POKE 559,62) or double line resolution (POKE 559,46). Our program illustrated here demonstrates only double line resolution: single line resolution would cause some changes in the player start address, etc.

With a POKE 53277,3 you enable P/M graphics.

Now you can draw the player; lines 100 to 130 build player 0. Each player is eight bits wide:

7	6	5	4	3	2	1	0	Bit
128	64	32	16	8	4	2	1	del. value

If you want to draw a player you must calculate the bytes you want to set for each line.

(E. g., if you want to set the right two bytes of any line of the player, you only Poke the decimal value of 1 + 2 = 3 to the address where you want to store it).

The missiles are two bits wide and all lie on the same line.

M3		M2		M1		M0		Missile
7	6	5	4	3	2	1	0	Bit
128	84	32	16	8	4	2	1	del. value

To set the missile set only the bits of the missile.
The simulation of the flight of the missiles follows physical formulas (see lines 1000 to 2200). The lines of the program are explained in REM statements. Please look at them.

```
5 OPEN #1,4,0,"K:"
10 GRAPHICS 3+16:SETCOLOR 0,3,5:SETCOLOR 4,8
,2:REM BACKGROUNDCOLOR
12 COLOR 1:PLOT 0,23:DRAWTO 39,23:REM DRAW G
ROUND
20 X=240:X1=120:Y1=100:VX=150:V01=200:VY=0:G
=9.81:Y=20:REM INITIALISATION
40 J=PEEK(106)-8:REM P/M BASEADRESS
50 POKE 54279,J:REM STORE P/M ADRESS
60 PMBASE=256*J:REM PLAYERADDRESS
70 POKE 559,46:REM DOUBLE LINE RESOLUTION
80 POKE 53277,3:REM P/M ENABLE
100 FOR I=PMBASE+512 TO PMBASE+640:POKE I,0:
NEXT I:REM CLEAR PLAYER 0
110 RESTORE 130:FOR I=PMBASE+512+Y TO PMBASE
+517+Y:READ A:POKE I,A:NEXT I:REM THIS LOOP
DRAWS PLAYER 0
120 REM PLAYER 0 IN DATA
130 DATA 1,1,3,63,255,126
132 FOR I=PMBASE+640 TO PMBASE+768:POKE I,0:
NEXT I:REM CLEAR PLAYER 1
134 RESTORE 138:FOR I=PMBASE+640+Y1 TO PMBAS
E+647+Y1:READ A:POKE I,A:NEXT I:REM THIS LOO
P DRAWS PLAYER 1
138 DATA 1,2,36,56,56,60,62,63
140 POKE 53260,2:REM MISSILE SIZE
142 FOR I=PMBASE+768 TO PMBASE+896:POKE I,0:
NEXT I:REM CLEAR PLAYER 2
144 RESTORE 148:FOR I=PMBASE+768+Y1 TO PMBAS
E+774+Y1:READ A:POKE I,A:NEXT I:REM THIS LOO
P DRAWS PLAYER 2
148 DATA 16,40,0,101,74,32,8
150 FOR I=PMBASE+384 TO PMBASE+510:POKE I,0:
NEXT I:REM CLEAR MISSILES (1+2);ONE MISSILE
IS 8 BIT WIDE
```

```
160 POKE 704,88:REM PLAYER/MISSILE 0:PINK
165 POKE 705,197:REM PLAYER/MISSILE 1:GREEN
167 POKE 706,210:REM PLAYER2:DARK
169 POKE 707,200:REM PLAYER 3:YELLOW
190 POKE 53278,15:FOR X=200 TO 44 STEP -2:PO
KE 53248,X:REM '53278' := COLLISION CLEAR; L
OOP: PLAYER MOVEMENT
200 ST=STICK(0):X1=X1+3*((ST=7)+(X1<200)):X1
=X1-3*((ST=11)+(X1>50)):REM CALCULATION OF N
EW PLAYER 1 POSITION
210 POKE 53249,X1:REM DRAW PLAYER AT NEW POS
ITION
220 IF PEEK(764)<>255 AND GUN<>1 THEN GET #1
,W:W=(W-48)/8:VX1=VO1*COS(W):VY1=VO1*SIN(W):
T1=0:GUN=1:XP1=X1+6
225 REM LINE 220 ASKS KEYBOARD WHETHER ANY K
EY IS PRESSED DOWN
230 IF GUN=1 THEN GOSUB 2000:REM MOVEMENT OF
 MISSILE 1
240 IF BOMB=1 THEN GOSUB 1000:GOTO 260:REM M
OVEMENT OF MISSILE 0
250 IF RND(0)<0.5 THEN BOMB=1:XP=X:T=0:Z=50:
REM BOMB TO BE DROPPED ?
260 NEXT X:GOSUB 5000:GOTO 190:REM AGAIN AND
 AGAIN
302 IF I=20 THEN POKE 53258,3
1000 REM MISSILE 0 FIRED
1005 TRAP 1200:REM IF MISSILE OUT OF RANGE T
HEN G.1200
1040 Z=Z-1:T=T+1:REM TIME INCREASED BY 1
1150 YO=G/2*T*T:YO=(YO/50)+Y+4:REM MATH ROUT
INE TO CALCULATE YVECTOR OF FLIGHT OF MISSIL
E
1160 IF YO>110 THEN BOMB=0:SOUND 0,0,0,0:GOS
UB 3000:RETURN :REM REM MISSILE REACHED GROU
ND
1170 XM=(VX*T)/50:REM CALCULATION OF XVECTOR
 OF FLIGHT
1180 POKE PMBASE+384+YH,0:POKE PMBASE+384+YO
,3:POKE 53252,XP-XM:SOUND 0,Z,10,14:YH=YO:RE
M MISSILE FLIGHT
1190 RETURN :REM END OF SUBRUTINE
```

```
1200 POKE PMBASE+384+Y0,0:BOMB=0:SOUND 0,0,0
,0:RETURN
2000 REM MISSILE 1 FIRED
2010 TRAP 2200:REM IF MISSILE IS OUT OF RANG
E THEN G.2200
2140 T1=T1+1:REM TIME INCREASED BY 1
2150 Y1=-G/2*T1*T1+VY1*T1:Y1=ABS((Y1/20)-105
):REM CALCULATION OF YVECTOR OF FLIGHT
2160 IF Y1<10 OR Y1>120 THEN GUN=0:SOUND 1,0
,0,0:RETURN :REM MISSILE OUT OF RANGE?
2170 XM1=(VX1*T1)/20:REM CALCULATION OF XVEC
TOR OF FLIGHT
2180 POKE PMBASE+384+YH1,0:POKE PMBASE+384+Y
1,4:POKE 53253,XP1+XM1:YH1=Y1:SOUND 1,T1,8,1
4:REM MISSILE FLY
2190 GOSUB 3500:RETURN :REM END OF SUBROUTIN
E
2200 POKE PMBASE+384+Y1,0:GUN=0:SOUND 1,0,0,
0:RETURN
3000 IF PEEK(53256)<>3 THEN RETURN :REM IF N
O COLLISION THEN RETURN
3005 SCOREBOMBER=SCOREBOMBER+1:IF SCOREBOMBE
R=10 THEN 6000:REM SCORE OF BOMBER
3007 POKE 53249,250:POKE 53250,X1:REM PLAYER
 1 OUT OF VIEW;PLAYER 2 AT OLD-PLAYER 1-POSI
TION
3010 FOR I=1 TO 30:SOUND 0,V,8,14:FOR T=1 TO
 10:NEXT T:V=INT((250-180)*RND(0)*180):REM E
XPLOSION
3020 IF I=10 THEN POKE 53258,1:REM CHANGE SI
ZE OF PLAYER 2
3022 IF I=20 THEN POKE 53258,3:REM CHANGE SI
ZE OF PLAYER 2
3040 NEXT I:POKE 53258,0:POKE 53250,250
3050 SOUND 0,0,0,0:GUN=0:BOMB=0:GOTO 190:REM
 NO SOUND;NEW ATTACK
3500 IF PEEK(53257)<>1 THEN RETURN :REM IF N
O COLLISION THEN RETURN
3502 GOSUB 7000
3505 SCOREGUN=SCOREGUN+1:IF SCOREGUN=10 THEN
 GOTO 6000:REM SCORE
```

```
3507 POKE 53248,250:POKE 53251,X:REM PLAYER
O OUT OF VIEW;PLAYER 3 AT OLD-PLAYER O-POSIT
ION
3510 FOR I=1 TO 30:SOUND 0,V,8,14:FOR T=1 TO
 10:NEXT T:V=INT((250-180)*RND(0)*180):REM E
XPLOSION
3520 IF I=10 THEN POKE 53259,1:REM CHANGE SI
ZE OF PLAYER 3
3522 IF I=20 THEN POKE 53259,3:REM CHANGE SI
ZE OF PLAYER 3
3540 NEXT I:POKE 53259,0:POKE 53251,250
3550 SOUND 1,0,0,0:GUN=0:BOMB=0:GOTO 190
5000 Y=INT((20-80)*RND(0)+80):REM NEW VRTICA
L POSITION
5003 FOR I=PMBASE+512 TO PMBASE+640:POKE I,0
:NEXT I:REM CLEAR PLAYER 0
5005 RESTORE 5020:FOR I=PMBASE+512+Y TO PMBA
SE+517+Y:READ A:POKE I,A:NEXT I:REM THIS LOO
P DRAWS PLAYER 0
5020 REM PLAYER 0 IN DATA
5030 DATA 1,1,3,63,255,126
5040 RETURN
5300 STOP
6000 PRINT "SCORE OF GUN: ";SCOREGUN:PRINT :
PRINT "SCORE OF BOMBER: ";SCOREBOMBER:END
7000 FOR I=PMBASE+896 TO PMBASE+1024:POKE I,
0:NEXT I:REM CLEAR PLAYER 3
7010 RESTORE 7020:FOR I=PMBASE+896+Y TO PMBA
SE+903+Y:READ A:POKE I,A:NEXT I:REM THIS LOO
P DRAWS PLAYER 3
7020 DATA 4,8,2,73,0,42,0,82
7030 RETURN
```

ROBOT ATTACK

ROBOT ATTACK is a game which shows the easy use of the player-missile graphics. All four players are used. Player 0 thru 2 are the robots and player 3 is the gun.

Program Structure:

Line 100 – 180:	Writing of the titles.
Line 190 – 200:	Wait until a key is pressed.
Line 220:	Turn off the cursor.
Line 230:	GOTO subroutine. Initialize fast vertical movement of the player.
Line 240:	Set the parameters for speed control.
Line 250 – 270:	Initialice the screen.
Line 280:	Determine RAMTOP and set PM base address.
Line 290:	Turn on PM functions.
Line 310:	Erase all players and missiles.
Line 320:	Set the colors of the robots. (player 0–2)
Line 330–350:	Initialice the player 0–2
Line 360:	Set the color of player 3
Line 370:	Set horizontal position of player 3
Line 380 – 400:	Initialize player 3
Line 410:	Store the horizontal and vertical position of the missile.
Line 420:	Store horizontal and vertical position of player 0–2.
Line 430:	GOTO subroutine MOVING.
Line 440 – 480:	Control of the movement of player 3.
Line 490:	Shoot missile when button is pressed.
Line 500:	Jump back. (infinite loop)
Line 520 – 680:	Initialize fast vertical movement of the player.
Line 690 – 940:	Subroutine. Moves the player 0 – 2 and the missile 3. Collision detection and sound.
Line 950 –1020:	You lost, the robots are invaded. Start for a new game.

You can extend the game. Let the robot shoot and accelerate the movement of the gun.

```
100 GRAPHICS 2+16
110 CLR
120 PRINT #6:PRINT #6:PRINT #6;"      ROBOT A
TTACK "
130 PRINT #6:PRINT #6:PRINT #6;" COPYRIGHT (
C) 1982"
140 PRINT #6
150 PRINT #6;"                  "
160 PRINT #6;"                    "
170 PRINT #6;"                       "
180 PRINT #6;"
190 IF PEEK(764)=255 THEN 190
200 POKE 764,255
210 GRAPHICS 0
220 POKE 752,1
230 GOSUB 520:GOSUB 610
240 SCH1=2:SCH2=3
250 ? "}"
260 DIM X(4)
270 SETCOLOR 2,0,0
280 A=PEEK(106)-8:POKE 54279,A:PMBASE=256*A
290 POKE 559,46:POKE 53277,3:REM ENABLE PM
300 RESTORE 350
310 FOR I=PMBASE+384 TO PMBASE+1023:POKE I,0
:NEXT I:REM EREASE  PM
320 POKE 704,216:POKE 705,216:POKE 706,216:R
EM SET COLOR PM 0-2
330 REM SET PLAYER 0-2
340 FOR J=512 TO 894 STEP 128:RESTORE 350:Y=
INT(RND(0)*90)+10:FOR I=PMBASE+J+Y TO PMBASE
+J+9+Y:READ A:POKE I,A:NEXT I:NEXT J
350 DATA 24,24,56,63,60,60,60,24,24,60
360 POKE 707,56
370 POKE 53251,200
380 RESTORE 400:YP=65
390 FOR I=PMBASE+896+YP TO PMBASE+900+YP:REA
D D:POKE I,D:NEXT I
400 DATA 28,28,124,28,28
410 MX=0:MY=0
420 X(1)=0:X(2)=-20:X(3)=-35
430 GOSUB 690
440 REM GUN MOVEMENT
450 A=STICK(0)
```

```
460 IF A=13 THEN FOR I=1 TO SCH2:D=USR(DOWN,
PMBASE+895+YP):YP=YP+1:NEXT I:GOTO 490
470 IF A=14 THEN FOR I=1 TO SCH2:D=USR(UP,PM
BASE+895+YP):YP=YP-1:NEXT I:GOTO 490
480 FOR I=1 TO 15:NEXT I
490 IF STRIG(0)=0 THEN IF MX=0 THEN MX=200:M
Y=YP:POKE PMBASE+386+MY,255
500 GOTO 430
510 STOP
520 DIM UPCODE$(22):UP=ADR(UPCODE$)
530 RESTORE 580
540 FOR I=UP TO UP+20
550 READ BYTE:POKE I,BYTE
560 NEXT I:RETURN
570 REM
580 DATA 104,104,133,204,104,133,203
590 DATA 160,1,177,203,136,145,203
600 DATA 200,200,192,7,208,245,96
610 DIM DOWNCODE$(22):DOWN=ADR(DOWNCODE$)
620 RESTORE 660
630 FOR I=DOWN TO DOWN+20
640 READ BYTE:POKE I,BYTE
650 NEXT I:RESTORE :RETURN
660 DATA 104,104,133,204,104,133,203
670 DATA 160,6,177,203,200,145,203
680 DATA 136,136,192,255,208,245,96
690 FOR J=1 TO SCH1
700 FOR I=1 TO 3
710 POKE 53278,17:REM COLLISION CLEAR
720 SOUND 1,12,8,9
730 X(I)=X(I)+2
740 IF X(I)<0 THEN 770
750 IF X(I)>200 THEN 950
760 POKE 53247+I,X(I)
770 SOUND 1,0,0,0
780 IF MX=0 THEN 810
790 IF MX<40 THEN 880
800 MX=MX-2:POKE 53255,MX
810 D=PEEK(53259)
820 IF D=0 OR D=8 THEN 920
830 IF D=4 THEN D=3
840 SOUND 0,60,10,8:SOUND 1,70,10,8:SOUND 2,
10,8,2
```

```
850 X(D)=0
860 PUNKT=PUNKT+2
870 IF INT(PUNKT/28)=PUNKT/28 THEN SCH2=SCH2
-1
880 MX=0:POKE 53255,MX
890 POKE PMBASE+386+MY,0
900 POSITION 10,0:? "SCORE:  ";PUNKT;
910 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUND 2,0,0,
0
920 NEXT I
930 NEXT J
940 RETURN
950 FOR T=1 TO 10:SOUND 0,6,13,8:SOUND 1,0,0
,0
960 FOR X=1 TO 18:NEXT X:SOUND 0,0,0,0:SOUND
 1,11,13,8:NEXT T
970 SOUND 0,0,0,0
980 FOR Z=30 TO 200:SOUND 2,Z,10,8:NEXT Z
990 FOR S=1 TO 6:FOR P=60 TO 40 STEP -2:SOUN
D 0,P,10,8:FOR X=1 TO 10:NEXT X
1000 NEXT P:FOR P=40 TO 60 STEP 2:SOUND 0,P,
10,8:FOR X=1 TO 10:NEXT X:NEXT P:NEXT S
1010 SOUND 0,0,0,0
1020 FOR I=1 TO 8:POKE 53247+I,0:NEXT I:GOTO
 100
```

BALL

BALL

This game demonstrates the use of player-missile graphics as you may need if you want to program games like Tennis, Squash, Wallbreaker etc. A ball is moving in a playfield that you can try to catch. If the ball bumps into your racket, you will get points.

Lines 10–99 initiate the playfield, the player and the missile.
Line 100 checks if the joystick is pressed.
Line 119 always clears the collision register.
Line 120 clears the last position of the missile.
Lines 121 + 122 set the vectors of the missile (the movement of the missile is the result of X-vector and Y-vector. In case of a collision with the wall only one vector will be changed).
Line 130 draws the new missile position.
Line 140: In case of collision with your racket the score is increased by one.

This program demonstrates only the basic programming techniques, you need to create a game with a ball being reflected at a wall.
Try to program "Tennis", "Squash", or "Wallbreaker".
Make the game quicker by using greater steps in the movement, or even machine language subroutines.

```
10 GRAPHICS 5:SETCOLOR 2,0,0
15 COLOR 1:PLOT 0,0:DRAWTO 79,0:DRAWTO 79,39
:DRAWTO 0,39:DRAWTO 0,0:REM PLAYFIELD
20 POKE 559,46:REM SINGLE LINE RES.
40 POKE 704,188:REM PLAYER:PINK
50 I=PEEK(106)-8:REM P/M ADRESS
60 POKE 54279,I:REM STORE ADRESS
70 POKE 53277,3:REM ENABLE P/M
75 POKE 53256,1:REM PLAYER TWICE NORMAL SIZE
```

```
80 PMBASE=I*256
85 FOR I=PMBASE+512 TO PMBASE+640:POKE I,0:N
EXT I:REM CLEAR PLAYER
90 POKE PMBASE+512+85,60:POKE PMBASE+513+85,
60:REM DRAW PLAYER
92 FOR I=PMBASE+384 TO PMBASE+512:POKE I,0:N
EXT I:REM CLEAR MISSILE
94 POKE PMBASE+384,3:REM DRAW MISSILE
98 YV=INT(1*RND(0)+1):XV=INT(-4*RND(0)+2):RE
M STARTVECTORS FOR MISSILE
99 YM=19:XM=120:REM MISSILE TOP OF FIELD
100 ST=STICK(0):X=X+2*((ST=7)+(X<195)):X=X-2
*((ST=11)+(X>45)):REM JOYSTICK??
110 POKE 53248,X
119 POKE 53278,0:REM COLLISION CLEAR
120 POKE PMBASE+384+YM1,0:REM CLEAR LAST POS
. OF MISSILE
121 XM=XM+XV:YM1=YM:YM=YM+YV:IF XM<50 OR XM>
205 THEN XV=-XV:SOUND 0,20,10,15:SOUND 0,0,0
,0:GOTO 120
122 IF YM<19 OR YM>92 THEN YV=-YV:SOUND 0,20
,10,15:SOUND 0,0,0,0:GOTO 120
130 POKE PMBASE+384+YM,3:POKE 53252,XM:REM D
RAW NEW MISSILE POSITION
140 IF PEEK(53256)<>0 THEN P=P+1:PRINT "}SCO
RE ";P:GOTO 92:REM 'IF COLLISION THEN PRINT
SCORE
200 GOTO 100:REM AGAIN AND AGAIN
```

SMART

SMART

SMART is a game in which the computer stores a combination of 6 different colors in 4 places. Each color may appear more than one time. You have to find out the correct colors and combinations in less than 10 trials. After each trial, the computer tells you, how many colors are in the right place (black dots) and how many colors are present, but in the wrong place (red dots). Using this information, you must correct this cast.

The program:

The program runs under ATARI BASIC or CP/A Basic. It uses graphic mode 11 with the video controller GTIA (an upgraded CTIA). With this chip it is possible to show 16 different colors in BASIC.

Structure of the program:

Line 100 – 141: Title and Copyright

Line 145: Wait until a key is pressed. Reset of the input buffer, otherwise the character is part of the next INPUT-instruction.

Line 200–225: Drawing of the playfield using the XIO FILL instruction.

Line 230–240: Draw details in the playfield

Line 350–364: The RND-function is used for creating 4 different colors. The numbers of the colors are stored in the array C (1–4)

Line 400–450: Read the joystick, which controls the color by vertical movement. Pressing the button sets current color to cursor position.

Line 600–628: The computer checks how many colors can be marked with black (SCH) or red (WEI).

Line 640–662: Show the result to the player.

Line 670 – 700: "You lost". Show the right colors.

Line 710 – 730: "New game?"

Line 800 – 810: "You won"

Line 1000–1030:	Subroutine, mark empty cursor position with predefined color. Move the cursor to next position.
Line 2000:2050:	Change the color.
Line 3000–3050:	Look for the coordinates of the next free cursor position (# PL) of the trial (# VER).
Line 4000–4040:	Draw the result of one trial.

Note:
Colors may be different on different ATARI computers (PAL-version in Europe, NTSC in the US etc.).

```
90 CLR :REM CLEAR MEMORY
100 GRAPHICS 2+16
110 PRINT #6:PRINT #6;"       SMART       ":PRIN
T #6:PRINT #6
120 PRINT #6;"COPYRIGHT (C)1982 BY";
130 PRINT #6;" ELCOMP PUBLISHING, INC. ";
135 PRINT #6;"    53 REDROCK LANE      ";
140 PRINT #6;"   POMONA, CA 91766      ";
145 REM KEY PRESSED ?
150 DU=PEEK(764):IF DU=255 THEN 145
160 POKE 764,255:REM CLEAR INPUT
200 GRAPHICS 11:REM TURN ON GRAPHICS 11
210 COLOR 1:REM DRAW IN BRIGHT ORANGE (GOLD)
215 REM DRAW PLAYING FIELD IN COLOR 1
220 PLOT 38,156:DRAWTO 38,0:DRAWTO 0,0:POSIT
ION 0,156:POKE 765,1
225 XIO 18,#6,0,0,"S:"
230 REM DRAW HIDDEN FIELD
232 COLOR 0:REM DRAW IN BLACK
233 FOR I=2 TO 8 STEP 2
235 PLOT 1,I:DRAWTO 26,I
240 NEXT I
250 REM DRAW HOLES
255 COLOR 0:REM DRAW IN BLACK
260 FOR I=20 TO 130 STEP 12
270 FOR J=5 TO 20 STEP 4
280 FOR K=0 TO 3
282 PLOT J,I+K
284 NEXT K
286 NEXT J
```

```
288 NEXT I
289 COLOR 10:REM DRAW IN BLUE GREEN
290 FOR I=19 TO 131 STEP 12
292 FOR J=0 TO 4 STEP 4
294 PLOT 30,I+J:PLOT 34,I+J:PLOT 30,I+J+1:PL
OT 34,I+J+1
296 NEXT J
298 NEXT I
300 REM DRAW ALL EXISTING COLORS
310 FOR I=10 TO 120 STEP 20
312 READ F
314 FOR J=1 TO 10
316 COLOR F
318 PLOT 50,I+J:PLOT 51,I+J
320 NEXT J
322 NEXT I
340 DATA 3,5,7,10,13,14
350 REM CREATE COMBINATION
351 DIM A(4),B(4),C(I)
352 FOR I=1 TO 4
354 X=INT(RND(0)*6)
356 RESTORE 340
358 FOR J=1 TO X
360 READ Y
361 C(I)=Y
362 NEXT J
364 NEXT I
400 REM LOOK FOR COMBINATION,USE JOYSTICK #1
405 PL=1:VER=1:FA=1
410 IF FLAG=0 THEN COLOR 1:GOSUB 1000:FLAG=1
:GOTO 430
420 COLOR 0:GOSUB 1000:FLAG=0
430 IF STICK(0)=13 THEN FA=FA+1:GOSUB 2000
440 IF STICK(0)=14 THEN FA=FA-1:GOSUB 2000
450 IF STRIG(0)<>0 THEN 410
500 REM PLACE COLOR
510 GOSUB 2000
515 B(PL)=X
520 GOSUB 3000
530 FOR I=-4 TO 4
532 PLOT X-1,Y+I:PLOT X,Y+I:PLOT X+1,Y+I
534 NEXT I
540 IF PL=4 THEN 600
```

```
550 PL=PL+1
560 GOTO 410:REM LOOK FOR NEXT COMBINATION
600 REM COMBINATION RIGHT ?
602 FOR I=1 TO 4
604 A(I)=C(I)
606 NEXT I
608 SCH=0:WEI=0
610 FOR I=1 TO 4
612 IF B(I)=A(I) THEN SCH=SCH+1:B(I)=100:A(I
)=200
614 NEXT I
620 FOR I=1 TO 4
622 FOR J=1 TO 4
624 IF A(I)=B(J) THEN WEI=WEI+1:B(J)=100:A(I
)=200
626 NEXT J
628 NEXT I
640 IF SCH=4 THEN 800
650 AN=1
652 IF SCH<>0 THEN SCH=SCH-1:COLOR 0:GOSUB 4
000:GOTO 652
654 IF WEI<>0 THEN WEI=WEI-1:COLOR 4:GOSUB 4
000:GOTO 654
660 VER=VER+1
662 IF VER<11 THEN PL=1:GOTO 410:REM NEXT TR
IAL
670 FOR I=10 TO 200:SOUND 1,I,10,7:NEXT I:FO
R I=1 TO 200:NEXT I:SOUND 1,0,0,0
680 RESTORE 3015
682 FOR I=1 TO 4
684 READ X
686 COLOR C(I)
688 FOR K=0 TO 10
690 PLOT X-1,1+K:PLOT X,1+K:PLOT X+1,1+K
692 NEXT K
694 NEXT I
698 FOR I=10 TO 200:SOUND 1,I,10,7:NEXT I:FO
R I=1 TO 200:NEXT I:SOUND 1,0,0,0
700 FOR I=1 TO 500:NEXT I
710 ? "}}ONE MORE (1/0) ";
715 INPUT DU
720 IF DU=0 THEN ? "}";:END
730 GOTO 90
```

```
800 ? "YOU'RE A CHAMPION !"
810 GOTO 710
1000 REM DRAW EMPTY HOLE
1010 GOSUB 3000
1020 FOR K=0 TO 3
1022 PLOT X,Y+K
1024 NEXT K
1030 RETURN
2000 IF FA>6 THEN FA=1
2010 IF FA<1 THEN FA=6
2015 RESTORE 340
2020 FOR I=1 TO FA
2022 READ X
2024 NEXT I
2026 COLOR X
2030 FOR I=1 TO 10
2032 PLOT 60,I+10:PLOT 61,I+10
2034 NEXT I
2040 SOUND 1,100,12,7
2042 FOR I=1 TO 50:NEXT I
2044 SOUND 1,0,0,0
2050 RETURN
3000 REM CALCULATE COORDINATES FOR NEXT FREE
 PLACE
3010 RESTORE 3015
3015 DATA 5,9,13,17
3020 FOR I=1 TO PL
3022 READ X
3024 NEXT I
3030 Y=8
3040 FOR I=1 TO VER
3042 Y=Y+12
3044 NEXT I
3050 RETURN
4000 REM CALCULATE AND DISPLAY RESULT
4010 RESTORE 4015
4015 DATA 30,19,34,19,30,23,34,23
4020 FOR I=1 TO AN
4022 READ X,Y
4024 NEXT I
4026 AN=AN+1
4030 PLOT X,Y+(VER-1)*12
4032 PLOT X,Y+(VER-1)*12+1
4040 RETURN
```

BARRIER

BARRIER is a game for one or two players. It is similar to "surround". The object is to force your opponent (the other player or the computer) to stop. The players are moving squares that leave behind a wall that neither player can go through. In addition, the computer places 2 or more barriers on the playing field.

Three graphics modes are available: 3, 5, or 7. In mode three, the graphics are coarse, but the game is very fast. In mode 5 the graphics are finer and the game slower. In mode 7 the graphics are very fine and the game rather slow.

```
5 K=3:UU=1:HIN=2:KUKUCK=17
10 DIM A$(12),B$(12),C$(12),D$(12)
12 GOTO 8999
15 POP :POKE 752,0:SOUND 0,0,0,0:SOUND 1,0,0
,0:SOUND 2,0,0,0:SOUND 3,0,0,0:GOTO KUKUCK
17 N=0:M=0
20 GRAPHICS 0:FOR W=0 TO 79:? "";:NEXT W:? "
   *>>> B A R R I E R <<<*    ";
23 FOR W=0 TO 79:? "";:NEXT W
25 SEL=30:GOTO 6000
30 SEL=40:GOTO 6060
40 SEL=50:GOTO 6030
50 IF UU=1 THEN SEL=60:GOTO 6090
60 SEL=30000
65 IF UU=1 THEN G=5500:EE=5000:Z=300:S=3000
66 IF UU<>1 THEN S=4000:EE=410
70 A=0:B=0:C=0:D=0:AA=0:BB=0:CC=0:DD=0:GH=0:
HJ=XYZ:ABC=0
80 IF K=7 THEN XYZ=K+10:QQ=159:WW=79:R=76:Y=
82:T=40:GOTO 100
90 IF K=3 THEN XYZ=6:R=18:T=10:Y=20:QQ=39:WW
=19:GOTO 100
95 XYZ=8:K=5:R=36:T=20:Y=42:QQ=79:WW=39
```

```
100 GRAPHICS K:COLOR 3:SETCOLOR 2,0,0
120 PLOT 0,0:DRAWTO QQ,0:DRAWTO QQ,WW:DRAWTO
 0,WW:DRAWTO 0,0
140 SETCOLOR 2,1,13
200 SETCOLOR 4,9,4
220 SETCOLOR 1,0,0
225 SETCOLOR 0,10,8
230 COLOR 3:SETCOLOR 2,1,13
240 COLOR 0:SETCOLOR 4,9,4
245 GOSUB 9100
250 COLOR 2:PLOT R,T
255 COLOR 1:PLOT Y,T
257 FOR JJ=1 TO 50:NEXT JJ
260 BALLA=265:GOTO 30005
265 G=5500
270 W=1100
280 X=1150
290 J=15
300 HH=15:COLOR 2
320 GOTO 980
330 LOCATE AA+R,BB+T,YY
335 IF YY<>0 THEN GOTO S
340 COLOR 2:PLOT AA+R,BB+T
400 COLOR 1
405 GOTO EE
410 IF J<>15 THEN 1010
420 GOTO 1000
440 LOCATE CC+Y,DD+T,YY
460 IF YY<>0 THEN GOTO 2000
470 COLOR 1:PLOT CC+Y,DD+T
480 FOR QW=1 TO 1:LLL=485:GOTO 490:NEXT QW
485 GOTO 620
490 HH=15:J=15
520 IF STICK(0)=15 AND STICK(1)=15 THEN 600
540 HH=STICK(0):Z=300
560 J=STICK(1)
600 GOTO LLL
620 IF HH=15 THEN 300
640 GOTO 982
980 IF STICK(0)=15 AND HH=15 THEN GOTO W
982 IF STICK(0)=14 OR HH=14 THEN W=1100:GOTO
 W
984 IF STICK(0)=7 OR HH=7 THEN W=1120:GOTO W
```

```
986 IF STICK(0)=13 OR HH=13 THEN W=1140:GOTO
 W
988 IF STICK(0)=11 OR HH=11 THEN W=1160:GOTO
 W
990 GOTO W
1000 IF STICK(1)=15 THEN GOTO X
1010 IF STICK(1)=14 OR J=14 THEN X=1110:GOTO
 X
1030 IF STICK(1)=7 OR J=7 THEN X=1130:GOTO X
1050 IF STICK(1)=13 OR J=13 THEN X=1150:GOTO
 X
1070 IF STICK(1)=11 OR J=11 THEN X=1170:GOTO
 X
1080 GOTO X
1100 BB=BB-1:GOTO 330
1110 DD=DD-1:GOTO 440
1120 AA=AA+1:GOTO 330
1130 CC=CC+1:GOTO 440
1140 BB=BB+1:GOTO 330
1150 DD=DD+1:GOTO 440
1160 AA=AA-1:GOTO 330
1170 CC=CC-1:GOTO 440
2000 ? A$;" wins."
2020 M=M+1
2500 ? "SCORE IS  ";M;" : ";N
2520 GOTO 30000
2600 GOTO 2560
3000 ? "The computer wins"
3010 FOR Q=1 TO 500:NEXT Q
3020 N=N+1
3040 GOTO 2500
4000 ? C$;" wins"
4010 FOR Q=1 TO 500:NEXT Q
4020 N=N+1
4040 GOTO 2500
5000 Z=620:COLOR 1:A=C:B=D:F=A:H=B:I=0:E=0
5020 GOSUB G:LOCATE A+Y,B+T,YY:V=G
5040 IF YY<>0 THEN 5140
5060 C=A:D=B:E=1:GOSUB G
5080 LOCATE A+Y,B+T,YY
5100 IF YY<>0 THEN LLL=5140:GOTO 490
5105 ABC=ABC+1
5110 GH=GH+1:IF GH=HJ THEN GH=0:ABC=ABC*1.2+
0.6:HJ=INT(3*K+ABC*K*RND(0)):GOTO 5140
```

```
5120 PLOT C+Y,D+T:GOTO Z
5140 IF G=5500 OR G=5520 THEN OO=INT(RND(0)*
2+3)
5160 IF G=5540 OR G=5560 THEN OO=INT(RND(0)*
2+1)
5180 A=F:B=H:ON OO GOSUB 5500,5520,5540,5560
5200 LOCATE A+Y,B+T,YY:IF YY<>0 THEN LLL=530
0:GOTO 490
5220 C=A:D=B:GOSUB G:LOCATE A+Y,B+T,YY:L=G
5240 IF YY<>0 THEN I=1:LLL=5300:GOTO 490
5260 PLOT C+Y,D+T:GOTO Z
5300 A=F:B=H:ON OO GOSUB 5520,5500,5560,5540
5320 LOCATE A+Y,B+T,YY:IF YY<>0 THEN LLL=538
0:GOTO 490
5340 PLOT A+Y,B+T:C=A:D=B:GOTO Z
5380 IF I=1 THEN A=F:B=H:GOSUB L:PLOT A+Y,B+
T:C=A:D=B:GOTO Z
5400 IF E=1 THEN A=F:B=H:GOSUB V:PLOT A+Y,B+
T:C=A:D=B:GOTO Z
5420 GOTO 2000
5500 B=B-1:G=5500:RETURN
5520 B=B+1:G=5520:RETURN
5540 A=A-1:G=5540:RETURN
5560 A=A+1:G=5560:RETURN
6000 ? "graphic mode 3,5 or 7 (";K;") ";:TRA
P 6010:INPUT K:GOTO SEL
6010 ? ;K:GOTO SEL
6030 ? "number of players "
6035 ? "1 or 2 (";UU;") ";:TRAP 6040:INPUT U
U:GOTO 7000
6040 ? UU:GOTO SEL
6060 ? "number of barriers (";HIN;") ";:TRAP
 6070:INPUT HIN:GOTO SEL
6070 ? HIN:GOTO SEL
6090 ? "enter name of 1st player (";A$;") ";
:INPUT A$:IF A$="" THEN 6110
6100 B$=A$:GOTO 6120
6110 A$=B$:? A$
6120 IF UU=2 THEN 6150
6130 GOTO SEL
6150 ? "enter name of 2nd player (";C$;") ";
:INPUT C$:IF C$="" THEN 6170
6160 D$=C$:GOTO SEL
```

```
6170 ? D$:C$=D$:GOTO SEL
7000 IF UU=2 THEN 6090
7010 GOTO SEL
8999 GRAPHICS 2
9000 POKE 764,255:FOR W=0 TO 83:? #6;"*";:NE
XT W:POKE 752,1
9010 ? #6;"  BARRIER  ";:FOR W=0 TO 84:? #6;
"*";:NEXT W
9020 FOR WAS=0 TO 15
9030 FOR WOL=15 TO 0 STEP -1
9032 COL=COL+1
9035 SOUND 0,COL*1.1,14,10:SOUND 1,ABS(COL-2
70),14,10
9037 SOUND 2,RND(0)*WAS+5*4,10,10:SOUND 3,CO
L*WAS,10,10
9040 SETCOLOR 0,WAS,WOL:SETCOLOR 4,WOL,WAS
9045 POSITION 5,9:? #6;INT(COL);"  ";WAS;":"
;WOL;"  "
9047 GOSUB 9447
9049 FOR LOCK=0 TO 2:NEXT LOCK
9050 NEXT WOL:NEXT WAS
9055 GOTO 9300
9060 COL=RND(0)*COL
9070 GOTO 9020
9100 IF HIN=0 THEN 9120
9110 COLOR 3:FOR W=1 TO HIN:PLOT RND(0)*(QQ-
4)+2,RND(0)*(WW-4)+2:DRAWTO RND(0)*(QQ-4)+2,
RND(0)*(WW-4)+2:NEXT W
9120 RETURN
9300 FOR JM=0 TO 14 STEP 1.5:FOR DER=0 TO 6:
SOUND 0,150+JM*4,6,ABS(JM-14)
9305 GOSUB 9447
9310 SOUND 3,100+JM*5,4,ABS(JM-14)
9320 SOUND 1,230+JM*2,10,ABS(JM-14)
9330 SOUND 2,RND(0)*220,4,ABS(JM-14)
9335 SETCOLOR 0,RND(0)*15,DER:SETCOLOR 4,DER
,RND(0)*15:NEXT DER
9340 NEXT JM:GOTO 9060
9447 IF PEEK(764)<>255 THEN 15
9448 IF STRIG(0)=0 OR STRIG(1)=0 THEN 15
9450 RETURN
25620 FOR W=0 TO 2:SOUND 0,58,14,14:NEXT W:S
OUND 0,0,0,0:RETURN
```

```
30000 BALLA=65
30005 ? "waiting..."
30010 POKE 764,255
30030 G=PEEK(764)
30040 IF STRIG(0)=0 OR STRIG(1)=0 THEN GOTO
BALLA
30045 IF G=255 THEN 30030
30050 GOSUB 25620:IF G=33 THEN G=VOR
30100 POKE 764,255:? :? :VOR=G
30200 IF G=61 THEN 6000
30210 IF G=23 THEN 6030
30220 IF G=35 THEN 6090
30230 IF G=57 THEN 6060
30240 IF G=50 THEN M=0:N=0:GOTO 2500
30250 IF G=21 THEN KUKUCK=65:GOTO 12
30260 IF G=42 THEN 20
30270 IF G=40 THEN 17
31000 GOTO 30010
```

KNIGHT-BATTLE

KNIGHT-BATTLE is a game for two players. One is the castle-lord, the other is the fiendish knight. Both players try to destroy the other by firing on him with artillery shells. You must destroy your opponent, before he destroys you. To fire, you input the angle (0–90^0) and the amount of powder (1–15 bags). Watch the wind, as it will affect the trajectory of your shots.
Good luck!

```
10 DIM BURG$(15),RIT$(15),C$(15),Z$(3),FR(3)
20 GRAPHICS 2+16:SETCOLOR 4,3,0
30 PRINT #6
40 PRINT #6
50 PRINT #6
60 PRINT #6
70 PRINT #6;"   KNIGHT-BATTLE"
80 IF PEEK(764)=255 THEN 80
90 POKE 764,255
95 GOSUB 9000
100 GRAPHICS 7
102 REM DRAWING OF HILL
105 REM
110 COLOR 1:GOSUB 6000:B=79
120 PLOT 0,79:DRAWTO 159,79:PLOT 0,78:DRAWTO
159,78
130 FOR I=50 TO 90
140 C=INT(RND(0)*7):IF C=4 THEN C=3
142 IF C=5 OR C=6 THEN C=0
143 IF B<21 THEN C=0
145 B=B-C
150 PLOT I,B:DRAWTO I,79:NEXT I
155 FOR O=70 TO 79:PLOT 90,O:DRAWTO 159,O:NEX
T O
```

```
160 A=79:FOR I=130 TO 90 STEP -1
170 C=INT(RND(0)*6):IF C=4 THEN C=0
172 IF C=5 THEN C=3
175 IF B>=A-2 THEN 190
180 A=A-C
190 PLOT I,A:DRAWTO I,79
200 NEXT I
210 REM DRAWING OF CASTLE
220 COLOR 3
225 REM BODY OF CASTLE
230 PLOT 155,69:DRAWTO 155,55:DRAWTO 127,55:P
OSITION 127,69:POKE 765,3:XIO 18,#6,0,0,"S:"
240 N1=127:M1=55:N2=133:GOSUB 260
250 N1=149:N2=155:GOSUB 260
255 GOTO 270
260 PLOT N2,M1:DRAWTO N2,M1-12:DRAWTO N1,M1-1
2:POSITION N1,M1:POKE 765,3:XIO 18,#6,0,0,"S:
"
265 RETURN
270 REM WINDOWS OF CASTLE
280 S1=133:COLOR 1
290 FOR I=1 TO 4
300 PLOT S1,63:DRAWTO S1,63-2:DRAWTO S1-2,63-
2:DRAWTO S1-2,63:DRAWTO S1,63
310 PLOT S1-1,63-1:S1=S1+6:NEXT I
320 REM ROOFS OF CASTLE
330 COLOR 1:H1=152:R=1:GOSUB 340
335 COLOR 2:N1=127:N2=133:H1=130:R=2:GOSUB 34
0
337 GOTO 360
340 PLOT N2,M1-12:DRAWTO H1,M1-17:DRAWTO H1,M
1-17:POSITION N1,M1-12:POKE 765,R:XIO 18,#6,0
,0,"S:"
350 RETURN
360 REM POINTS ON CASTLE
370 COLOR 3:FOR I=132 TO 149 STEP 4:PLOT I,54
:PLOT I+1,54:NEXT I
380 REM DRAWING OF THE OPPONENT
385 COLOR 2
390 PO=INT(RND(0)*39+5)
400 PLOT PO+6,77:DRAWTO PO+3,77-5:DRAWTO PO+3
,77-5:POSITION PO,77:POKE 765,2:XIO 18,#6,0,0
,"S:"
```

```
405 WD=INT(RND(0)*30+1)
410 REM BURG$: INPUT
415 C9=0:GOSUB 1040
417 PRINT "CASTLE-LORD ";BURG$;": ";:GOSUB 10
00
420 H4=130:H5=42
425 REM
427 IF C9=1 THEN H4=P0+3:H5=8:GOSUB 1040:PRIN
T "KNIGHT ";RIT$;": ";:GOSUB 1000
430 T=0
434 REM CALCULATION : BURG$
435 GRAPHICS 7+32:GOSUB 6000:DEG :T=0
436 IF C9=1 THEN 450
440 WINK=180-WINK
450 V=PULV+2+WW
460 X=V*T*COS(WINK)
470 Y=V*T*SIN(WINK)-0.5*9.8*T^2
472 IF Y1<2 THEN 480
475 COLOR 0:PLOT X1,Y1
480 X1=H4+V*X:Y1=79-(H5+V*Y)
481 IF X1>159 OR X1<1 OR Y1>79 THEN 520
482 IF Y1<2 THEN 500
485 FOR I=0 TO 2:LOCATE X1,Y1-I,FD:FR(I)=FD:I
F FR(I)=2 AND X1>110 THEN 3000:REM SELF DESTR
UCTION
486 IF FR(I)=2 THEN 4000:REM OPPONENT DESTROY
ED
487 NEXT I:IF FR(0)=3 OR FR(0)=1 THEN 600
490 COLOR 2:PLOT X1,Y1:SOUND 0,Y1,8,9
500 T=T+0.05:GOTO 460
520 X=0:Y=0:X1=0:Y1=0
530 IF C9=0 THEN C9=1:GOTO 425
540 GOTO 410
600 DATA 1,1,1,0,-1,0,-1,1,0,0,2,0,2,2,2,1,2,
-1,-1,-1
605 COLOR 0:SOUND 0,0,0,0:SOUND 1,160,0,15
610 FOR T1=1 TO 10:READ Z1,Z2:PLOT X1+Z1,Y1+Z
2:PLOT X1+Z2,Y1+Z1:NEXT T1:RESTORE
620 FOR B1=1 TO 200:NEXT B1:SOUND 1,0,0,0:GOT
O 520
1000 REM BURG$: INPUT OF ANGLE
1005 TRAP 1005:PRINT "ENTER THE ANGLE ";:INPU
T WINK
```

```
1010 REM BURG$: INPUT OF POWDER
1020 TRAP 1020:PRINT "ENTER POWDER ";:INPUT P
ULV:IF PULV<1 OR PULV>15 THEN 1020
1030 RETURN
1040 SOUND 0,0,0,0:REM WIND
1050 FF=INT(RND(0)*2+1)
1055 NN=INT(RND(0)*2+1):HH=RND(0)*7
1056 IF NN=1 THEN WD=WD+HH:GOTO 1060
1057 WD=WD-HH:IF ABS(WD)>60 THEN 1050
1060 WD=ABS(WD):WD=INT(WD):GF=ABS(WD):IF FF=1
 THEN PRINT "}              WIND: < ";GF;" ":G
OTO 1080
1070 PRINT "}              WIND: > ";GF;" ":W
D=-WD
1080 IF C9=1 THEN WD=-WD
1100 WW=WD/50:SOUND 3,30,8,ABS(WD)/8:RETURN
3000 I=2:NEXT I:GOSUB 15000:REM DESTROY CASTL
E
3010 GRAPHICS 7+16+32:GOSUB 6000:FOR I=1 TO 2
5:SETCOLOR 2,3,14:SETCOLOR 4,7,0:GOSUB 7000
3020 C7=RND(0)*6+127
3025 C5=RND(0)*6+127
3030 C8=RND(0)*15+55-17
3035 C6=RND(0)*15+55-17:SETCOLOR 4,3,2
3040 COLOR 0:PLOT C7,C8:PLOT C5,C6:SETCOLOR 2
,3,0:GOSUB 7000:NEXT I
3100 GOSUB 7040
3500 FOR I=55-17 TO 55-17+12
3510 FOR P1=127 TO 127+6
3520 COLOR 0:PLOT P1,I:NEXT P1:NEXT I
3530 FOR I=1 TO 200:C7=RND(0)*(D1+17)+127:D1=
D1+0.06:C8=RND(0)*(D1+3)+55:PLOT C7,C8:NEXT I
3540 GOSUB 7050:FOR I=127 TO 159:PLOT 127,60:
DRAWTO I,37:NEXT I
3550 FOR I=127 TO 148:PLOT 127,65:DRAWTO I,54
:NEXT I:GOSUB 15000:GOSUB 6000
3560 FOR I=1 TO 500:NEXT I:GRAPHICS 0:IF C9=1
 THEN 3610
3565 GOSUB 15000:GOSUB 16000:PRINT "
     BRAVO":GOSUB 8000
3570 PRINT "";BURG$;", YOU'RE A FINE LORD !"
3580 PRINT "YOU HAVE BLASTED YOURSELF AND THE
 ";INT(RND(0)*1000+100)
```

```
3590 PRINT "RESIDENTS OF YOUR CASTLE"
3600 GOTO 10000
3610 GOSUB 15000:GOSUB 16000:GOSUB 8000:PRINT
 "BRAVO ";RIT$;" !YOU HAVE BLASTED THE CASTLE
 AND"
3620 PRINT INT(RND(0)*1000+100);" RESIDENTS."
:GOSUB 15000:GOTO 10000
4000 I=2:NEXT I:GRAPHICS 7+16+32:GOSUB 6000
4005 FOR I=1 TO 25:SETCOLOR 4,7,0
4010 C7=RND(0)*7+P0:SETCOLOR 4,3,2
4020 C8=RND(0)*5+72:GOSUB 7000
4030 PLOT C7,C8:NEXT I:GOSUB 7050
4031 COLOR 0:FOR I=0 TO 5:PLOT P0,72+I:DRAWTO
 P0+7,72+I:NEXT I:FOR U6=1 TO 200:NEXT U6:GOS
UB 15000:GOSUB 6000
4040 FOR I=1 TO 500:NEXT I:GRAPHICS 0:IF C9=1
 THEN 4090
4050 GOSUB 15000:GOSUB 16000:PRINT "
      BRAVO !":GOSUB 8000
4060 PRINT "LORD ";BURG$;" YOU HAVE DESTROYED
"
4070 PRINT "YOUR OPPONENT ! CONGRATULATIONS !
":GOTO 10000
4090 GOSUB 15000:GOSUB 16000:GOSUB 8000:PRINT
 "          B R A V O !!!"
4095 PRINT "KNIGHT ";RIT$;" YOU HAVE DESTROYE
D YOUR"
4100 PRINT "OWN CAMP ! NO ONE SURVIVED.":GOTO
 10000
6000 SETCOLOR 4,9,11:SETCOLOR 0,15,0:SETCOLOR
 2,12,4:SETCOLOR 1,1,14:RETURN
7000 SOUND 0,180,0,8
7010 FOR B2=0 TO 5:SOUND 1,B2,14,8:NEXT B2
7020 SOUND 1,0,0,0:RETURN
7040 ZZ=8
7045 SOUND 1,170,16,ZZ:SOUND 2,200,2,ZZ:RETUR
N
7050 ZZ=15:GOSUB 7045:SOUND 3,145,8,ZZ:RETURN

8000 FOR I=150 TO 81 STEP -1:SOUND 0,I,10,8:N
EXT I:SOUND 1,60,14,8:SOUND 2,96,14,8
8010 FOR B2=1 TO 200:NEXT B2:GOSUB 15000:RETU
RN
```

```
9000 GOSUB 16000:PRINT "            KNIGHT - B
ATTLE"
9140 PRINT "ENTER NAME OF THE CASTLE-LORD ";:
INPUT BURG$
9150 PRINT "ENTER NAME OF THE KNIGHT ";:INPUT
 RIT$
9160 RETURN
10000 TRAP 10000:PRINT "NEW GAME ";:INPUT Z$:
IF Z$(1,1)="N" THEN END
10010 C$=BURG$:BURG$=RIT$:RIT$=C$:X1=0:Y1=0:D
1=0:GOTO 100
15000 FOR B2=0 TO 3:SOUND B2,0,0,0:NEXT B2:RE
TURN
16000 GRAPHICS 0:SETCOLOR 4,4,2:SETCOLOR 2,4,
2:RETURN
```

CALENDAR

CALENDAR

This program will produce a calendar for one year. Leap years are included. The result can be output to either the screen or to a printer.

```
10 REM --- CALENDARPROGRAM
20 REM --- THE PROGRAM CALCULATES THE
    CALENDAR OF ONE YEAR
30 DIM A(150),A1(366),A$(10),B$(12),C$(2):OP
EN #2,4,0,"K:"
40 GRAPHICS 2:SETCOLOR 4,8,2:SETCOLOR 2,8,2:
POSITION 2,3:? #6;"CALENDARPROGRAM"
100 ? "for which year do you want the
 calendar ";:INPUT J
105 ? "}output to screen (1) or printer (2)"
::GET #2,AU
107 IF AU=50 THEN LPRINT CHR$(14);"
calendar for ";J:LPRINT :LPRINT :LPRINT ""
110 J1=J:IF J<100 THEN J=J+1900
120 M=1:B$=" ":L=0:IF INT(J/4)<>J/4 THEN 150
130 IF INT(J/100)=J/100 THEN IF INT(J/400)<>
J/400 THEN 150
140 B$="(leap-year)":L=1
150 T=INT((J-1893)/4)
160 C=(J-1893+T)/7:C=INT((C-INT(C))*7+0.5)
165 RESTORE 1390:FOR I=1 TO 12:READ A:A(I)=A
:NEXT I:A(2)=A(2)+L
170 FOR M=1 TO 12:RESTORE 1400:FOR I=1 TO M:
READ A$:NEXT I
180 D=0:IF M=1 THEN 200
190 FOR I=1 TO M-1:D=D+A(I):NEXT I
200 D=(D+C)/7
205 D=INT((D-INT(D))*7+0.5)
210 IF D=0 THEN D=7
220 REM
```

```
230 T=0:FOR I=D-2 TO 0 STEP -1:T=T+1:A(T+12)
=(-I):NEXT I
240 P=1:FOR J=D TO 7:A(J+12)=P:P=P+1:NEXT J
250 RESTORE 1420:W=0:IF AU=50 THEN 400
260 GRAPHICS 2:SETCOLOR 4,8,2:FOR J=0 TO 18
STEP 3:READ B$:POSITION J,0:? #6;B$:NEXT J
290 FOR I=2 TO 9
300 P=0:FOR J=0 TO 18 STEP 3
310 P=P+1:Q=A(P+12)+W*7
320 IF Q<=0 AND AU=50 THEN LPRINT CHR$(14);"
     ";:GOTO 370
330 IF Q<=0 OR Q>A(M) THEN 370
334 IF AU=50 AND Q<10 THEN LPRINT CHR$(14);"
 ";
335 IF AU=50 THEN LPRINT CHR$(14);Q;"   ";:G
OTO 370
340 POSITION J,I:? #6;Q
370 NEXT J:W=W+1:IF AU=50 THEN LPRINT
375 NEXT I:IF AU=50 THEN NEXT M:RUN
380 ? "}";A$;:GET #2,K:NEXT M:RUN
400 LPRINT CHR$(14);A$:LPRINT :LPRINT :FOR J
=1 TO 7:READ C$:LPRINT CHR$(14);C$;"   ";:NE
XT J:LPRINT :GOTO 290
1390 DATA 31,28,31,30,31,30,31,31,30,31,30,3
1
1400 DATA JANUARY,FEBRUARY, MARCH,APRIL,MAY,
JUNE,JULY,AUGUST,SEPTEMBER,OCTOBER,NOVEMBER,
DECEMBER
1420 DATA SU,MO,TU,WE,TH,FR,SA
```

GUNFIGHT

GUNFIGHT is for two players. Each player uses one joystick. Plug the joystick in port 1 and 2. Port 1 refers to the left player, port 2 to the right player.
GUNFIGHT starts with the Copyright-statement. Press one of the two trigger buttons and the playfield appears. Now you can move the cowboys.

The Game
You shoot by pressing the trigger. Each player has 10 cowboys and each cowboy 50 bullets. By holding the button pressed the gun keeps firing until the cowboy has no more bullets. There are 2 ways to be killed: Your opponent can shoot you or you can drown in the sea.
After each round the game waits for you to press the trigger. The game-status appears. It shows you the actual number of cowboys and bullets. When you press the trigger the game goes on with the next round. The winner of the game will be shown after you press the trigger button. Pressing the trigger the game starts again with the Copyright-statement.

The Sea
Don' t touch this blue area because you will be drowned.

If you run out of bullets:
If you don' t have any bullets you can only flee and hope that the enemy will touch the sea.

You can hide your cowboy behind any one of the three rocks. But when your opponent also hides behind a rock, he can shoot you as long as his gun is behind the rock.

The program starts at hex 1000. To key the program in, you need a machine language monitor or the ATARI® Editor/Assembler cartridge.
The starting address of the program GUNFIGHT is 1016 hex.
Start with GOTO 1016. Skip the location between 175F and 1F98.

1F98	=	Start of display list
2000	=	End of display list and beginning of the background. Type in everything like it is. (Use the fill-command of a machine language monitor).
2FFF		End of background

```
**************************************
*                                    *
*        GUNFIGHT FOR ATARI          *
*                                    *
*        BY HANS-CHRISTOPH WAGNER    *
*                                    *
*        07-12-1981 NIJMEGEN-NL      *
*                                    *
**************************************

COUT     EQU $F6A4          SCREENOUT
PRIOR    EQU $26F           PRIORITIES
CONSOL   EQU $D01F          SPEAKER
HITCLR   EQU $D01E          CLR COLL.
DLISTL   EQU $230           DISP. LIST POINTER
DLISTH   EQU $231
PFCOL0   EQU $2C4           PLAYFIELDCOLOR 0
PFCOL1   EQU $2C5           PLAYFIELDCOLOR 1
PFCOL2   EQU $2C6           PLAYFIELDCOLOR 2
PFCOL3   EQU $2C7           PLAYFIELDCOLOR 3
PFCOL4   EQU $2C8           PLAYFIELDCOLOR 4
PCOLR0   EQU $2C0           PLAYERCOLOR 1
PCOLR1   EQU $2C1           PLAYERCOLOR 2
SDMCTL   EQU $22F           DMA CONTROLLER
AUDF1    EQU $D200          AUDIOFREQUENCE 1
AUDC1    EQU $D201          AUDIOCONTROL 1
AUDF2    EQU $D202          AUDIOFREQUENCE 2
AUDC2    EQU $D203          AUDIOCONTROL 2

HPOSM0   EQU $D004          HOR. POS. MISSILE 1
HPOSM1   EQU $D005          HOR. POS. MISSILE 2
HPOSP0   EQU $D000          HOR. POS. PLAYER 1
HPOSP1   EQU $D001          HOR. POS. PLAYER 2

M0PF     EQU $D000          COLL. MISSILE 1 TO PLAYF.
M0PL     EQU $D008          COLL. MISSILE 1 TO PLAYER
M1PF     EQU $D001          COLL. MISSILE 2 TO PLAYF.
M1PL     EQU $D009          COLL. MISSILE 2 TO PLAYER
P0PF     EQU $D004          COLL. PLAYER 1 TO PLAYF.
P0PL     EQU $D00C          COLL. PLAYER 1 TO PLAYER
P1PF     EQU $D005          COLL. PLAYER 2 TO PLAYF.
P1PL     EQU $D00D          COLL. PLAYER 2 TO PLAYER
STICK0   EQU $278           JOYSTICK 1
STICK1   EQU $279           JOYSTICK 2
STRIG0   EQU $284           TRIGGER 1
STRIG1   EQU $285           TRIGGER 2

DEC1     EQU PTX+1          AUX. DECIMAL CONVERSION
WINFL    EQU $D1            SHOWS WINNER
BULLC0   EQU $D2            BULLETS PLAYER 1
BULLC1   EQU $D3            BULLETS PLAYER 2
LIFEC0   EQU $D4            COWBOYS PLAYER 1
LIFEC1   EQU $D5            COWBOYS PLAYER 2
SPMBASE  EQU $D6            PLAYER MISSILE DMA BASE
LASTFL   EQU $D7            AUX. FLAG
SWTCHFL  EQU $D8            FLAG TO INDICATE LEFT ,RIGHT SHOOTING
LEGXFL   EQU $D9            GENERAL MOVE LEGFL
LEG0FL   EQU $DA            MOVE LEGFL. PL. 1
LEG1FL   EQU $DB            MOVE LEGFL. PL. 2
LEGC0    EQU $DC            COUNTER TO KNOW WHEN HAS TO MOVE LEGS
```

```
                LEGC1     EQU $DD
                SCOUNTO   EQU $DE           COUNTER TO SLOW DOWN SO. 1
                SCOUNT1   EQU $DF           COUNTER TO SLOW DOWN SO. 2
                PTO       EQU $E0           PM POINTER OF PL. 1
                PT1       EQU $E2           PM POINTER OF PL. 2
                PTX       EQU $E4           GENERAL PM POINTER
                MPTO      EQU $E6           PM POINTER OF M. 1
                MPT1      EQU $E8           PM POINTER OF M. 2
                MOCOUNT   EQU $EA           COUNTER TO SLOW DOWN M.1
                M1COUNT   EQU $EB           COUNTER TO SLOW DOWN M.2
                SONDOFL   EQU $ED           INDICATES SOUND CREATION
                SOND1FL   EQU $EE
                MOX       EQU $EF           SHDW. HOR. POS. M1
                M1X       EQU $F0           SHDW. HOR. POS. M2
                COUNTO    EQU $F1           SLOW DOWN PL.1
                COUNT1    EQU $F3           SLOW DOWN PL.2
                JUMP      EQU $F5.7         JSR THROUGH THIS JMP INSTRUCTION
                SHP0H     EQU $F8           SHDW. HOR. POS. P1
                SHP1H     EQU $F9           SHDW. HOR. POS. P2
                MOFLG     EQU $FA           INDICATES TRIGGER 1
                M1FLG     EQU $FB           INDICATES TRIGGER 2
                MOVMOFL   EQU $FC           IND. M1 IS MOVING
                MOVM1FL   EQU $FD           IND. M2 IS MOVING
                SAUDF1    EQU $FE           SHDW OF AUDF1
                SAUDF2    EQU $FF           SHDW OF AUDF2

                MEMSIZE   EQU 106           TOP OF RAM
                PMBASE    EQU $D407         DMA PM BASE

                MASK0     EQU $03           BITMASK TO CREATE M1
                MASK01    EQU $FC           BITMASK TO CLEAR M1
                MASK1     EQU $0C           BITMASK TO CREATE M2
                MASK11    EQU $F3           BITMASK TO CLEAR M2

                          ORG $1016,$A800

1016: A2FF      START     LDX #$FF          INIT. STACK
1018: 9A                  TXS
1019: A94C                LDA #$4C
101B: 85F5                STA JUMP          INIT. JUMP
101D: A56A                LDA MEMSIZE
101F: E910                SBC #16
1021: 85D6                STA SPMBASE       CALC. PMBASE
1023: A908                LDA #$08
1025: 8D6F02              STA PRIOR         GIVE PRIORITY SEE MANUAL
1028: E8                  INX
1029: 8EC602              STX PFCOL2
102C: 8EC802              STX PFCOL4        PLAYFIELDCOLORS:=0 (BLACK)
102F: 20AC13              JSR INIT          INITIALISE OTHER STUFF

                *         POLLING LOOP

1032: 200D11    LOOP2     JSR MOVP0         SERVICE P1
1035: 204711              JSR MOVP1         SERVICE P2
1038: 206F12              JSR MOVM0         SERVICE MO
103B: 20D712              JSR MOVM1         SERVICE M1
```

```
103E: 205612           JSR CHCKTRIG    SERVICE TRIGGERS
1041: 209C14           JSR CHCKCOLL    CHECK COLL.
1044: 203F13           JSR SOUND       SERVICE SOUNDS
1047: 208F13           JSR SWITCH      SERVICE RIGHT OR LEFT COWBOY
104A: 4C3210           JMP LOOP2       POLL AGAIN

104D: 20B610 SETUP0    JSR MOVP0X      MOVE POINTERS TO GENERAL
1050: A5DA             LDA LEG0FL
1052: 85D9             STA LEGXFL      MOVE LEGFL TO GENERAL
1054: 24D8             BIT SWTCHFL     WHICH IMAGE RIGHT OR LEFT
1056: 1003             BPL OTHER1
1058: 4C6F10           JMP SETUPX1     SETUP RIGHT ONE
105B: 4C8E10 OTHER1    JMP SETUPX2     SETUP LEFT  ONE

105E: 20AD10 SETUP1    JSR MOVP1X      SEE SETUP0
1061: A5DB             LDA LEG1FL
1063: 85D9             STA LEGXFL
1065: 24D8             BIT SWTCHFL
1067: 3003             BMI OTHER2
1069: 4C6F10           JMP SETUPX1
106C: 4C8E10 OTHER2    JMP SETUPX2

106F: A014   SETUPX1   LDY #19+1       RIGHT ONE
1071: A206             LDX #6
1073: 24D9   LOOP3     BIT LEGXFL
1075: 1005             BPL LEG2        SPREAD LEGS?
1077: BDE310           LDA LEGTAB1,X
107A: D003             BNE STORE  A.T. ALWAYS
107C: BDEA10 LEG2      LDA LEGTAB2,X
107F: 91E4   STORE     STA (PTX),Y
1081: 88               DEY
1082: CA               DEX
1083: 10EE             BPL LOOP3
1085: B9D510 LOOP31    LDA BODYTAB1,Y  SETUP BODYPART
1088: 91E4             STA (PTX),Y
108A: 88               DEY
108B: 10F8             BPL LOOP31
108D: 60               RTS

108E: A014   SETUPX2   LDY #19+1       SETUP LEFT SEE SETUPX1
1090: A206             LDX #6
1092: 24D9   LOOP031   BIT LEGXFL
1094: 1005             BPL LEG21
1096: BDFF10           LDA LEGTAB21,X
1099: D003             BNE STORE01
109B: BD0611 LEG21     LDA LEGTAB22,X
109E: 91E4   STORE01   STA (PTX),Y
10A0: 88               DEY
10A1: CA               DEX
10A2: 10EE             BPL LOOP031
10A4: B9F110 LOOP311   LDA BODYTAB2,Y
10A7: 91E4             STA (PTX),Y
10A9: 88               DEY
10AA: 10F8             BPL LOOP311
10AC: 60               RTS
```

```
10AD: A5E2   MOVP1X   LDA PT1       MOVE POINTER TO GENERAL POINTER
10AF: 85E4            STA PTX
10B1: A5E3            LDA PT1+1
10B3: 85E5            STA PTX+1
10B5: 60              RTS

10B6: A5E0   MOVP0X   LDA PT0       SEE MOVP1X
10B8: 85E4            STA PTX
10BA: A5E1            LDA PT0+1
10BC: 85E5            STA PTX+1
10BE: 60              RTS

10BF: 20AD10 CLRPL1   JSR MOVP1X    MOVPOINTER
10C2: 4CCB10          JMP CLRPLX    CLEAR PLAYER

10C5: 20B610 CLRPL0   JSR MOVP0X    SEE CLRPL1
10C8: 4CCB10          JMP CLRPLX

10CB: A014   CLRPLX   LDY #19+1     CLEAR 20 BYTES
10CD: A900            LDA #0
10CF: 91E4   LOOP4    STA (PTX),Y
10D1: 88              DEY
10D2: 10FB            BPL LOOP4
10D4: 60              RTS           PLAYER DOESNT EXIST ANYMORE

             *        DIFFERENT IMAGES LEGS AND BODIES

10D5: 18187E BODYTAB1 DFB $18,$18,$7E,$18,$38,$18,$08
10D8: 183818
10DB: 08
10DC: 1C1C1C          DFB $1C,$1C,$1C,$7C,$3C,$1C
10DF: 7C3C1C
10E2: 1C              DFB $1C

10E3: 080808 LEGTAB1  DFB $08,$08,$08,$08,$08,$38
10E6: 080838
10E9: 38              DFB $38

10EA: 080816 LEGTAB2  DFB $08,$08,$16,$16,$12,$72
10ED: 161272
10F0: 72              DFB $72

10F1: 18187E BODYTAB2 DFB $18,$18,$7E,$18,$1C,$18
10F4: 181C18
10F7: 103838          DFB $10,$38,$38,$38,$3E,$3C,$38
10FA: 383E3C
10FD: 38
10FE: 38              DFB $38

10FF: 101010 LEGTAB21 DFB $10,$10,$10,$10,$10,$1C,$1C
1102: 10101C
1105: 1C

1106: 101068 LEGTAB22 DFB $10,$10,$68,$68,$48,$4E,$4E
1109: 68484E
110C: 4E
```

```
110D: E6F1    MOVP0  INC COUNT0       SLOWDOWNCOUNTER+:=1
110F: A5F1           LDA COUNT0
1111: C930           CMP #$30
1113: D031           BNE RTS1         IF COUNT<>$30 NO SERVICE
1115: A900           LDA #0
1117: 85F1           STA COUNT0       COUNT:=0
1119: AD7802         LDA STICK0
111C: 203D12         JSR COMPARE      REQUEST STICK
111F: B025           BCS RTS1         C=1 => NO SERVICE
1121: B98111         LDA JMPTAB0,Y
1124: 85F6           STA JUMP+1
1126: B98211         LDA JMPTAB0+1,Y
1129: 85F7           STA JUMP+2
112B: 20F500         JSR JUMP         EXECUTE PLAYER MOVEMENT
112E: 204D10         JSR SETUP0       SETUP NEW PLAYER
1131: E6DC           INC LEGC0        LEGC+:=1
1133: A5DC           LDA LEGC0
1135: C908           CMP #$08
1137: D00D           BNE RTS1         IF LEGC=8 THEN LEGS:=NOT (LEGS)
1139: A200           LDX #0
113B: 8E1FD0         STX CONSOL       MAKE NOISE
113E: 86DC           STX LEGC0        LEGC:=0
1140: CA             DEX
1141: 8A             TXA
1142: 45DA           EOR LEG0FL       LEG0FL:=NOT (LEG0FL)
1144: 85DA           STA LEG0FL
1146: 60      RTS1   RTS

1147: E6F3    MOVP1  INC COUNT1       SEE MOVP0
1149: A5F3           LDA COUNT1
114B: C930           CMP #$30
114D: D031           BNE RTS2
114F: A900           LDA #0
1151: 85F3           STA COUNT1
1153: AD7902         LDA STICK1
1156: 203D12         JSR COMPARE
1159: B025           BCS RTS2
115B: B99111         LDA JMPTAB1,Y
115E: 85F6           STA JUMP+1
1160: B99211         LDA JMPTAB1+1,Y
1163: 85F7           STA JUMP+2
1165: 20F500         JSR JUMP
1168: 205E10         JSR SETUP1
116B: E6DD           INC LEGC1
116D: A5DD           LDA LEGC1
116F: C908           CMP #$08
1171: D00D           BNE RTS2
1173: A200           LDX #0
1175: 8E1FD0         STX CONSOL
1178: 86DD           STX LEGC1
117A: CA             DEX
117B: 8A             TXA
117C: 45DB           EOR LEG1FL
117E: 85DB           STA LEG1FL
1180: 60      RTS2   RTS
```

```
1181: A111AF JMPTAB0  DFW UP0,DOWN0,RIGHT0,LEFT0
1184: 11BD11
1187: CA11
1189: D711DD          DFW UR0,UL0,DR0,DL0
118C: 11E311
118F: E911

1191: EF11FD JMPTAB1  DFW UP1,DOWN1,RIGHT1,LEFT1
1194: 110B12
1197: 1812
1199: 25122B          DFW UR1,UL1,DR1,DL1
119C: 123112
119F: 3712

11A1: 20C510 UP0      JSR CLRPL0       CLRPLAYER
11A4: A5E0            LDA PT0
11A6: C90B            CMP #11
11A8: F004            BEQ NOMOV1       IF YCOOR=11 THEN NOMOVE
11AA: C6E0            DEC PT0          YCOOR-:=2
11AC: C6E0            DEC PT0
11AE: 60      NOMOV1  RTS

11AF: 20C510 DOWN0    JSR CLRPL0
11B2: A5E0            LDA PT0
11B4: C9E3            CMP #$E3
11B6: F004            BEQ NOMOV2       IF YCOOR=227 THEN NOMOVE
11B8: E6E0            INC PT0
11BA: E6E0            INC PT0          YCOOR+:=2
11BC: 60      NOMOV2  RTS

11BD: A4F8    RIGHT0  LDY SHPOH
11BF: C0C9            CPY #201         IF XCOOR>=200 THEN NOMOVE
11C1: B006            BCS RGTORTS
11C3: C8              INY
11C4: 8C00D0          STY HPOSP0
11C7: 84F8            STY SHPOH        XCOOR+:=1
11C9: 60      RGTORTS RTS

11CA: A4F8    LEFT0   LDY SHPOH
11CC: C032            CPY #50
11CE: 9006            BCC LFTORTS      IF XCOOR<50 THEN NOMOVE
11D0: 88              DEY
11D1: 8C00D0          STY HPOSP0
11D4: 84F8            STY SHPOH        XCOOR-:=1
11D6: 60      LFTORTS RTS

11D7: 20BD11 UR0      JSR RIGHT0       RIGHTMOVE
11DA: 4CA111          JMP UP0          UPMOVE

11DD: 20CA11 UL0      JSR LEFT0        LEFTMOVE
11E0: 4CA111          JMP UP0          UPMOVE

11E3: 20BD11 DR0      JSR RIGHT0       RIGHTMOVE
11E6: 4CAF11          JMP DOWN0        DOWNMOVE

11E9: 20CA11 DL0      JSR LEFT0        LEFTMOVE
11EC: 4CAF11          JMP DOWN0        DOWNMOVE
```

```
11EF: 20BF10 UP1      JSR CLRPL1        SEE UPO
11F2: A5E2            LDA PT1
11F4: C90B            CMP #$0B
11F6: F004            BEQ NOMOV3
11F8: C6E2            DEC PT1
11FA: C6E2            DEC PT1
11FC: 60      NOMOV3  RTS

11FD: 20BF10 DOWN1    JSR CLRPL1        SEE DOWNO
1200: A5E2            LDA PT1
1202: C9E3            CMP #$E3
1204: F004            BEQ NOMOV4
1206: E6E2            INC PT1
1208: E6E2            INC PT1
120A: 60      NOMOV4  RTS

120B: A4F9    RIGHT1  LDY SHP1H         SEE RIGHTO
120D: C0C9            CPY #201
120F: B006            BCS RGT1RTS
1211: C8              INY
1212: 8C01D0          STY HPOSP1
1215: 84F9            STY SHP1H
1217: 60      RGT1RTS RTS

1218: A4F9    LEFT1   LDY SHP1H         SEE LEFTO
121A: C032            CPY #50
121C: 9006            BCC LFT1RTS
121E: 88              DEY
121F: 8C01D0          STY HPOSP1
1222: 84F9            STY SHP1H
1224: 60      LFT1RTS RTS

1225: 200B12  UR1     JSR RIGHT1        SEE URO
1228: 4CEF11          JMP UP1

122B: 201812  UL1     JSR LEFT1         SEE ULO
122E: 4CEF11          JMP UP1

1231: 200B12  DR1     JSR RIGHT1        SEE DRO
1234: 4CFD11          JMP DOWN1

1237: 201812  DL1     JSR LEFT1         SEE DLO
123A: 4CFD11          JMP DOWN1

123D: A007    COMPARE LDY #7            THERE ARE 8 DIRECTIONS
123F: D94E12  LOOP5   CMP COMTAB,Y
1242: F005            BEQ FOUND         STICK HAS VALUE
1244: 88              DEY
1245: 10F8            BPL LOOP5         IF NOT ALL VALUES CHECKED
1247: 38              SEC               C=1 => NO VALUE
1248: 60              RTS
1249: 98      FOUND   TYA
124A: 0A              ASL
124B: A8              TAY               Y:=2*Y
124C: 18              CLC               C=0 => VALUE
124D: 60              RTS
```

```
124E: 0E0D07 COMTAB   DFB 14,13,7,11,6,10,5,9  DECIMAL OF 4BIT JOYSTICK
1251: 0B060A                                                VALUES
1254: 0509

1256: 38     CHCKTRIG SEC                 C=1
1257: AD8402          LDA STRIG0          IF TRGGER1<>0 THEN CHECK TRIGGER2
125A: D007            BNE CTRIG1
125C: A5D2            LDA BULLC0
125E: F003            BEQ CTRIG1          ELIF BULLETS<>0 THEN
1260: 66FA            ROR M0FLG           SET M0FLG VIA TRUE CARRY
1262: 38              SEC                 C=1
1263: AD8502 CTRIG1   LDA STRIG1          SEE ABOVE
1266: D006            BNE CTRGRTS
1268: A5D3            LDA BULLC1
126A: F002            BEQ CTRGRTS
126C: 66FB            ROR M1FLG
126E: 60     CTRGRTS  RTS

126F: 24FC   MOVM0    BIT MOVM0FL
1271: 3029            BMI MOVM01          IF MOVM0FL THEN SERVICE
1273: A5FA            LDA M0FLG
1275: 85FC            STA MOVM0FL         MOVM0FL:=M0FLG
1277: 85ED            STA SOND0FL         SOND0FL:=M0FLG
1279: 1047            BPL RTS3            IF NO TRIGGER THEN RETURN
127B: C6D2            DEC BULLC0          BULLETS-:=1
127D: A9AF            LDA #$AF
127F: 8D01D2          STA AUDC1           SET NOISE VOLUME ETC.
1282: A5E0            LDA PT0
1284: 85E6            STA MPT0            YCOOR(MISSILE):=YCOOR(PLAYER)
1286: A5F8            LDA SHP0H
1288: 24D8            BIT SWTCHFL
128A: 3003            BMI STORE1
128C: 18              CLC
128D: 6908            ADC #8
128F: 85EF   STORE1   STA M0X
1291: 8D04D0          STA HPOSM0          XCOOR(M):=IF RIGHTCOWBOY
              *                                     THEN XCOOR(P)+8
              *                                     ELSE XCOOR(P)
1294: A00A            LDY #10             YCOOR+:=10
1296: B1E6            LDA (MPT0),Y
1298: 0903            ORA #MASK0          SET MISSILE
129A: 91E6            STA (MPT0),Y
129C: E6EA   MOVM01   INC M0COUNT
129E: A5EA            LDA M0COUNT
12A0: C910            CMP #$10
12A2: D01E            BNE RTS3            IF M0COUNT <>16 NO SERVICE
12A4: A900            LDA #0
12A6: 85EA            STA M0COUNT         M0COUNT:=0
12A8: A5EF            LDA M0X
12AA: C9D0            CMP #208            IF XCOOR>207 THEN RESET MISSILE
12AC: B015            BCS RESM0
12AE: C92D            CMP #45             IF XCOOR<45 THEN REST MISSILE
12B0: 9011            BCC RESM0
12B2: 24D8            BIT SWTCHFL
12B4: 3005            BMI DECR
12B6: E6EF            INC M0X
12B8: 4CBD12          JMP STORE2
12BB: C6EF   DECR     DEC M0X             IF RIGHTCOWBOY THEN INCREMENT
              *                           ELSE DECREMENT
```

```
12BD: A5EF    STORE2   LDA MOX
12BF: 8D04D0           STA HPOSM0
12C2: 60      RTS3     RTS
12C3: 46FC    RESM0    LSR MOVM0FL        MOVM0FL:=FALSE
12C5: 46FA             LSR M0FLG          M0FLG:=FALSE
12C7: 205813           JSR RSOUND0        RESET SOUND
12CA: A00A             LDY #10            YCOOR:=YCOOR+11
12CC: B1E6             LDA (MPT0),Y
12CE: 29FC             AND #MASK01        CLEAR MISSILE
12D0: 91E6             STA (MPT0),Y
12D2: A900             LDA #0
12D4: 85EA             STA M0COUNT        M0COUNT:=0
12D6: 60               RTS

12D7: 24FD    MOVM1    BIT MOVM1FL        SEE MOVM0
12D9: 3029             BMI MOVM11
12DB: A5FB             LDA M1FLG
12DD: 85FD             STA MOVM1FL
12DF: 85EE             STA SOND1FL
12E1: 1047             BPL RTS4
12E3: C6D3             DEC BULLC1
12E5: A9AF             LDA #$AF
12E7: 8D03D2           STA AUDC2
12EA: A5E2             LDA PT1
12EC: 85E8             STA MPT1
12EE: A5F9             LDA SHP1H
12F0: 24D8             BIT SWTCHFL
12F2: 1003             BPL STORE3
12F4: 18               CLC
12F5: 6908             ADC #8
12F7: 85F0    STORE3   STA M1X
12F9: 8D05D0           STA HPOSM1
12FC: A00A             LDY #10
12FE: B1E8             LDA (MPT1),Y
1300: 090C             ORA #MASK1
1302: 91E8             STA (MPT1),Y
1304: E6EB    MOVM11   INC M1COUNT
1306: A5EB             LDA M1COUNT
1308: C910             CMP #$10
130A: D01E             BNE RTS4
130C: A900             LDA #0
130E: 85EB             STA M1COUNT
1310: A5F0             LDA M1X
1312: C9D0             CMP #208
1314: B015             BCS RESM1
1316: C92D             CMP #45
1318: 9011             BCC RESM1
131A: 24D8             BIT SWTCHFL
131C: 1005             BPL DECR1
131E: E6F0             INC M1X
1320: 4C2513           JMP STORE4
1323: C6F0    DECR1    DEC M1X
1325: A5F0    STORE4   LDA M1X
1327: 8D05D0           STA HPOSM1
132A: 60      RTS4     RTS
132B: 46FD    RESM1    LSR MOVM1FL
132D: 46FB             LSR M1FLG
132F: 208013           JSR RSOUND1
```

```
1332: A00A            LDY #10
1334: B1E8            LDA (MPT1),Y
1336: 29F3            AND #MASK11
1338: 91E8            STA (MPT1),Y
133A: A900            LDA #0
133C: 85EB            STA M1COUNT
133E: 60              RTS

133F: 24ED   SOUND    BIT SONDOFL
1341: 1024            BPL NOSONDO   IF NOSOUND1 THEN SERVICE SOUND2
1343: E6DE            INC SCOUNTO   SCOUNT+:=1
1345: A5DE            LDA SCOUNTO
1347: C910            CMP #$10
1349: D01C            BNE NOSONDO   IF SCOUNT<>16 THEN SERVICE SOUND2
134B: A900            LDA #0
134D: 85DE            STA SCOUNTO   SCOUNT:=0
134F: E6FE            INC SAUDF1    FREQUENCE:=FREQUENCE+1
1351: A5FE            LDA SAUDF1
1353: 8D00D2          STA AUDF1
1356: 100F            BPL NOSONDO
1358: A9A0   RSOUNDO  LDA #$A0      IF FREQUENCE=128 THEN RESET
135A: 8D01D2          STA AUDC1     VOLUME=0
135D: A900            LDA #0
135F: 85FE            STA SAUDF1    FREQUENCE=0
1361: 8D00D2          STA AUDF1
1364: 46ED            LSR SONDOFL   SONDOFL:=FALSE
1366: 60     RTS7     RTS
1367: 24EE   NOSONDO  BIT SOND1FL   SEE ABOVE
1369: 10FB            BPL RTS7
136B: E6DF            INC SCOUNT1
136D: A5DF            LDA SCOUNT1
136F: C910            CMP #$10
1371: D0F3            BNE RTS7
1373: A900            LDA #0
1375: 85DF            STA SCOUNT1
1377: E6FF            INC SAUDF2
1379: A5FF            LDA SAUDF2
137B: 8D02D2          STA AUDF2
137E: 10E6            BPL RTS7
1380: A9A0   RSOUND1  LDA #$A0
1382: 8D03D2          STA AUDC2
1385: A900            LDA #0
1387: 85FF            STA SAUDF2
1389: 8D02D2          STA AUDF2
138C: 46EE            LSR SOND1FL
138E: 60              RTS

138F: A5F8   SWITCH   LDA SHPOH
1391: C5F9            CMP SHP1H
1393: 66D8            ROR SWTCHFL   IF X(P1)>=X(P2) THEN SWTCHFL:=TRUE
1395: A5D8            LDA SWTCHFL
1397: 45D7            EOR LASTFL
1399: 1010            BPL RTS8      IF SWTCHFL=LASTFL THEN RETURN
139B: A5D8            LDA SWTCHFL   ELSE:
139D: 85D7            STA LASTFL    LASTFL:SWTCHFL
139F: 205E10          JSR SETUP1
13A2: 204D10          JSR SETUPO    SETUP NEW IMAGES
13A5: 20C312          JSR RESMO
13A8: 202B13          JSR RESM1     RESET MISSILES PREVENT SUICIDE
13AB: 60     RTS8     RTS
```

```
13AC: 202416 INIT     JSR PRINT
13AF: 7D              DFB $7D
13B0: 434F50          ASC "COPYRIGHT 1981 ING. W. HOFACKER GMBH"
13B3: 595249
13B6: 474854
13B9: 203139
13BC: 383120
13BF: 494E47
13C2: 2E2057
13C5: 2E2048
13C8: 4F4641
13CB: 434B45
13CE: 522047
13D1: 4D4248
13D4: 0D0D0D          DFB 13,13,13,13,13,13
13D7: 0D0D0D
13DA: 8D              DFB 141
13DB: 20C816          JSR SPACE
13DE: 202416          JSR PRINT
13E1: 47554E          ASC /GUNFIGHT/
13E4: 464947
13E7: 4854
13E9: 0D8D            DFB 13,141
13EB: 20C816          JSR SPACE
13EE: 202416          JSR PRINT
13F1: 425920          ASC "BY H.C. WAGNER"
13F4: 482E43
13F7: 2E2057
13FA: 41474E
13FD: 4552
13FF: 0D8D            DFB 13,141
1401: 200016          JSR STRTWAIT      WAIT FOR TRIGGER
1404: A97D            LDA #$7D
1406: 20A4F6          JSR COUT          CLEAR SCREEN
1409: 20DD16          JSR PLAYF         SETUP PLAYFIELD
140C: A93B            LDA #$3B
140E: 8DC002          STA PCOLR0
1411: A98A            LDA #$8A
1413: 8DC102          STA PCOLR1        SET PLAYERCOLORS
1416: A208            LDX #8
1418: A900            LDA #0
141A: 95E4   LOOP1    STA PTX,X
141C: CA              DEX
141D: 10FB            BPL LOOP1         INIT SOME 0-PAGE STUFF
141F: A90B            LDA #11
1421: 85E2            STA PT1
1423: A9E3            LDA #227
1425: 85E0            STA PT0           INIT PLAYER POSITION VERT.
1427: A5D6            LDA SPMBASE
1429: 18              CLC
142A: 6903            ADC #3
142C: 85E7            STA MPT0+1
142E: 85E9            STA MPT1+1
1430: 85E5            STA PTX+1
1432: 18              CLC
1433: 6901            ADC #1
1435: 85E1            STA PT0+1
1437: 18              CLC
1438: 6901            ADC #1
143A: 85E3            STA PT1+1
```

```
143C: 18              CLC
143D: 6903            ADC #3
143F: 85D7            STA LASTFL        CALCULATE PLM DMA ADRESSES
1441: A000            LDY #0
1443: 98              TYA
1444: 91E4   LOOP11   STA (PTX),Y
1446: C8              INY
1447: D0FB            BNE LOOP11
1449: E6E5            INC PTX+1
144B: A6E5            LDX PTX+1
144D: E4D7            CPX LASTFL
144F: 90F3            BCC LOOP11        CLEAR ALL PLAYERS
1451: A937            LDA #55
1453: 8D00D0          STA HPOSP0
1456: 85F8            STA SHP0H
1458: A9C8            LDA #200
145A: 8D01D0          STA HPOSP1
145D: 85F9            STA SHP1H         INIT HOR POS. PLAYERS
145F: 46FB            LSR M1FLG
1461: 46FA            LSR M0FLG
1463: A900            LDA #0
1465: 85DC            STA LEGC0
1467: 85DD            STA LEGC1
1469: 85D8            STA SWTCHFL
146B: 85D7            STA LASTFL        RESET SOME FLAGS AND COUNTERS
146D: A90A            LDA #10
146F: 85D4            STA LIFEC0
1471: 85D5            STA LIFEC1        EACH PLAYER 10 COWBOYS
1473: A932            LDA #50
1475: 85D2            STA BULLC0
1477: 85D3            STA BULLC1        EACH PLAYER 50 BULLETS
1479: A9CC            LDA #$CC
147B: 8DC502          STA PFCOL1
147E: A91F            LDA #$1F
1480: 8DC802          STA PFCOL4
1483: A90D            LDA #$0D
1485: 8DC402          STA PFCOL0
1488: A98C            LDA #$8C
148A: 8DC602          STA PFCOL2        GIVE PLAYFIELD COLORS
148D: A903            LDA #3
148F: 8D1DD0          STA 53277         START PLM DMA
1492: 204D10          JSR SETUP0
1495: 205E10          JSR SETUP1        SETUP PLAYERS
1498: 8D1ED0          STA HITCLR        RESET COLL.
149B: 60              RTS

149C: A9FF   CHCKCOLL LDA #$FF
149E: 85D1            STA WINFL         WINFL=TRUE
14A0: AD04D0          LDA P0PF
14A3: 49FF            EOR #$FF
14A5: 2D00D0          AND M0PF
14A8: 2901            AND #$01
14AA: F003            BEQ NOENDO        IF BEHIND ROCK NO RESET
14AC: 20C312          JSR RESM0         ELSE REST MISSILE
14AF: AD04D0 NOENDO   LDA P0PF
14B2: 2904            AND #$04
14B4: 0D09D0          ORA M1PL
14B7: 2905            AND #$05
14B9: F014            BEQ CHCKCOL1      IF DRUNK OR SHOT THEN KILL PL.
```

```
14BB: 202B13          JSR RESM1         RESET ALL MISSILES
14BE: 20C312          JSR RESMO
14C1: 20F716          JSR PLAYDOWN      PLAYER DOWN
14C4: A932            LDA #50
14C6: 85D2            STA BULLCO        BULLETS:=50
14C8: C6D4            DEC LIFECO        COWBOYS-:=1
14CA: D036            BNE DISP
14CC: 4C7416          JMP GAMEOVER      IF COWBOYS=0 THE GAMEOVER

14CF: A900   CHCKCOL1 LDA #0            SEE ABOVE
14D1: 85D1            STA WINFL
14D3: AD05D0          LDA P1PF
14D6: 49FF            EOR #$FF
14D8: 2D01D0          AND M1PF
14DB: 2901            AND #$01
14DD: F003            BEQ NOEND1
14DF: 202B13          JSR RESM1
14E2: AD05D0 NOEND1   LDA P1PF
14E5: 2904            AND #$04
14E7: 0D08D0          ORA MOPL
14EA: 2906            AND #$06
14EC: F039            BEQ RTS9
14EE: 20C312          JSR RESMO
14F1: 202B13          JSR RESM1
14F4: 20F716          JSR PLAYDOWN
14F7: A932            LDA #50
14F9: 85D3            STA BULLC1
14FB: C6D5            DEC LIFEC1
14FD: D003            BNE DISP
14FF: 4C7416          JMP GAMEOVER

1502: 20C510 DISP     JSR CLRPLO        INIT PLAYERS AGAIN
1505: 20BF10          JSR CLRPL1
1508: A90B            LDA #$0B
150A: 85E2            STA PT1
150C: A9E3            LDA #$E3
150E: 85E0            STA PTO
1510: A937            LDA #55
1512: 85F8            STA SHPOH
1514: 8D00D0          STA HPOSPO
1517: A9C8            LDA #200
1519: 85F9            STA SHP1H
151B: 8D01D0          STA HPOSP1
151E: 204D10          JSR SETUPO
1521: 205E10          JSR SETUP1
1524: 202B15          JSR SHOW          SHOW ACTUAL STATUS OF PLAYERS
1527: 8D1ED0 RTS9     STA HITCLR        RESET COLL
152A: 60              RTS

152B: 20CC15 SHOW     JSR RESTORE       SWITCH FROM PLAYFIELD TO TEXT
152E: 202416          JSR PRINT
1531: 0D              DFB 13
1532: 504C41          ASC "PLAYER 1 :"
1535: 594552
1538: 203120
153B: 3A
153C: 0D0D0D          DFB 13,13,13
153F: 42554C          ASC \BULLETS :\
```

```
1542: 4C4554
1545: 5320BA
1548: A5D2              LDA BULLC0
154A: 85E5              STA DEC1
154C: 204616            JSR DECOUT
154F: 202416            JSR PRINT
1552: 0D                DFB 13
1553: 434F57            ASC \COWBOYS :\
1556: 424F59
1559: 5320BA
155C: A5D4              LDA LIFEC0
155E: 85E5              STA DEC1
1560: 204616            JSR DECOUT
1563: 202416            JSR PRINT
1566: 0D0D0D            DFB 13,13,13
1569: 504C41            ASC "PLAYER 2 :"
156C: 594552
156F: 203220
1572: 3A
1573: 0D0D0D            DFB 13,13,13
1576: 42554C            ASC \BULLETS :\
1579: 4C4554
157C: 5320BA
157F: A5D3              LDA BULLC1
1581: 85E5              STA DEC1
1583: 204616            JSR DECOUT
1586: 202416            JSR PRINT
1589: 0D                DFB 13
158A: 434F57            ASC \COWBOYS :\
158D: 424F59
1590: 5320BA
1593: A5D5              LDA LIFEC1
1595: 85E5              STA DEC1
1597: 204616            JSR DECOUT
159A: 202416            JSR PRINT
159D: 0D8D              DFB 13,141
159F: 200016            JSR STRTWAIT
15A2: A97D              LDA #$7D
15A4: 20A4F6            JSR COUT
15A7: 20DD16            JSR PLAYF           CLEAR SCREEN SETUP PLAYF.
15AA: A93B              LDA #$3B
15AC: 8DC002            STA PCOLR0
15AF: A98A              LDA #$8A
15B1: 8DC102            STA PCOLR1
15B4: A98C              LDA #$8C
15B6: 8DC602            STA PFCOL2
15B9: A91F              LDA #$1F
15BB: 8DC802            STA PFCOL4
15BE: A9CC              LDA #$CC
15C0: 8DC502            STA PFCOL1
15C3: A903              LDA #3
15C5: 8D1DD0            STA 53277
15C8: 8D1ED0            STA HITCLR          INIT COLORS AND START DMA
15CB: 60                RTS

15CC: A900    RESTORE   LDA #0
15CE: 8D1DD0            STA 53277           DMA OF
15D1: 8DC602            STA PFCOL2
15D4: 8DC102            STA PCOLR1
15D7: 8DC002            STA PCOLR0
```

```
15DA: 8DC802          STA PFCOL4        RESET COLORS
15DD: A90A            LDA #$0A
15DF: 8DC502          STA PFCOL1        SET TEXT COLOR
15E2: A900            LDA #0
15E4: 8D2F02          STA SDMCTL        DMA CONTROL OFF
15E7: A559            LDA $59
15E9: 8D3102          STA DLISTH
15EC: A920            LDA #$20
15EE: 8D3002          STA DLISTL        SWITCH DISP. L. TO GR.0
15F1: A5D6            LDA SFMBASE
15F3: 8D07D4          STA FMBASE        I REALLY DON'T KNOW BUT IT WORKS
15F6: A93E            LDA #$3E
15F8: 8D2F02          STA SDMCTL        DMA CONTROLON
15FB: A97D            LDA #$7D
15FD: 4C6B16          JMP COUTO         RTS VIA CLEAR SCREEN


1600: 20C816 STRTWAIT JSR SPACE
1603: 202416          JSR PRINT
1606: 285452          ASC /(TRIGGER)/
1609: 494747
160C: 455229
160F: 8D              DFB 141
1610: 201B16 STRTWT1  JSR WAIT01        WAIT FOR NO TRIGGER
1613: AD8402 WAIT10   LDA STRIG0
1616: 2D8502          AND STRIG1
1619: D0F8            BNE WAIT10        WAIT FOR TRIGGER
161B: AD8402 WAIT01   LDA STRIG0
161E: 2D8502          AND STRIG1
1621: F0F8            BEQ WAIT01        WAIT FOR NO TRIGGER
1623: 60              RTS               RESULT:WAIT FOR TRIGGER EDGE

1624: 68      PRINT   PLA               THE VERY PRINT ROUTINE VIA STACK
1625: 85E4            STA PTX
1627: 68              PLA
1628: 85E5            STA PTX+1
162A: A200            LDX #0
162C: E6E4    LOOP23  INC PTX
162E: D002            BNE *+4
1630: E6E5            INC PTX+1
1632: A1E4            LDA (PTX,X)
1634: 297F            AND #$7F
1636: 206B16          JSR COUTO
1639: A200            LDX #0
163B: A1E4            LDA (PTX,X)
163D: 10ED            BPL LOOP23
163F: A5E5            LDA PTX+1
1641: 48              PHA
1642: A5E4            LDA PTX
1644: 48              PHA
1645: 60              RTS

              *       VERY ONE BYTE TO DECIMAL ROUTINE:

1646: A001    DECOUT  LDY #1
1648: A230    DECOUT0 LDX '0
164A: 38      DECOUT1 SEC
164B: A5E5            LDA DEC1
164D: F96916          SBC DECTAB,Y
```

```
1650: 9005            BCC DECOUT3
1652: 85E5            STA DEC1
1654: E8              INX
1655: D0F3            BNE DECOUT1
1657: 8A      DECOUT3 TXA
1658: 84E4            STY PTX
165A: 206B16          JSR COUTO
165D: A4E4            LDY PTX
165F: 88              DEY
1660: 10E6            BPL DECOUTO
1662: A5E5            LDA DEC1
1664: 0930            ORA '0
1666: 4C6B16          JMP COUTO

1669: 0A      DECTAB  DFB 10
166A: 64              DFB 100

166B: C90D    COUTO   CMP #13
166D: D002            BNE *+4
166F: A99B            LDA #$9B          IF CR THEN EOL ASCII TO ATASCII
1671: 4CA4F6          JMP COUT

1674: 205813 GAMEOVER JSR RSOUNDO
1677: 208013          JSR RSOUND1       RESET NOISE
167A: 20CC15          JSR RESTORE       RESTORE DISP. LIST TO GR.0
167D: 202416          JSR PRINT
1680: 0D0D0D          DFB 13,13,13,13,13,13,13,141
1683: 0D0D0D
1686: 0D8D
1688: 20C816          JSR SPACE
168B: 202416          JSR PRINT
168E: 47414D          ASC /GAME OVER/
1691: 45204F
1694: 564552
1697: 0D8D            DFB 13,141
1699: 20C816          JSR SPACE
169C: 202416          JSR PRINT
169F: 504C41          ASC \PLAYER \
16A2: 594552
16A5: A0
16A6: 24D1            BIT WINFL
16A8: 1004            BPL PLONE         IF WINFL THEN TWO ELSE ONE
16AA: A932            LDA '2
16AC: D002            BNE NUMOUT  A.T.  ALWAYS
16AE: A931    PLONE   LDA '1
16B0: 20A4F6 NUMOUT   JSR COUT
16B3: 202416          JSR PRINT
16B6: 205749          ASC / WINS!!!/
16B9: 4E5321
16BC: 2121
16BE: 0D0D0D          DFB 13,13,13,141
16C1: 8D
16C2: 200016          JSR STRTWAIT      WAIT TRIGGER EDGE
16C5: 4CAC13          JMP INIT          RTS VIA INIT


16C8: 202416 SPACE    JSR PRINT
16CB: 202020          ASC \
16CE: 202020
```

```
16D1: 202020
16D4: 202020
16D7: 202020
16DA: 20A0
16DC: 60              RTS

16DD: A900    PLAYF   LDA #0              SET DISPL. POINTER TO GR.7
                      *                   SET DMA CONTROL
16DF: 8D2F02          STA SDMCTL
16E2: A998            LDA #$98
16E4: 8D3002          STA DLISTL
16E7: A91F            LDA #$1F
16E9: 8D3102          STA DLISTH
16EC: A5D6            LDA SPMBASE
16EE: 8D07D4          STA PMBASE
16F1: A93E            LDA #$3E
16F3: 8D2F02          STA SDMCTL
16F6: 60              RTS

16F7: AD04D0 PLAYDOWN LDA P0PF
16FA: 0D05D0          ORA P1PF
16FD: 2904            AND #$04
16FF: F026            BEQ NODRUNK         IF NO SCOLL WITH SEA THEN DOWN
1701: 24D1            BIT WINFL
1703: 300A            BMI DRUNK1          WHO WAS THE WINNER
1705: 20BF10          JSR CLRPL1
1708: A982            LDA #130
170A: 8D01D0          STA HPOSP1
170D: D008            BNE DRUNK A.T.
170F: 20C510 DRUNK1   JSR CLRPL0
1712: A982            LDA #130
1714: 8D00D0          STA HPOSP0
1717: A980    DRUNK   LDA #128            PUT HAT IN SEA
1719: 85E4            STA PTX
171B: A002            LDY #2
171D: B9D510 DRNKLOOP LDA BODYTAB1,Y
1720: 91E4            STA (PTX),Y
1722: 88              DEY
1723: 10F8            BPL DRNKLOOP
1725: 302C            BMI TRIGWAIT  A.T. ALWAYS


1727: 24D1    NODRUNK BIT WINFL
1729: 3006            BMI PLAYDWN1        AGAIN WHO WAS THE WINNER
172B: 20AD10          JSR MOVP1X          MOVE POINTER
172E: 4C3417          JMP PLAYDWN         LETS GO DOWN
1731: 20B610 PLAYDWN1 JSR MOVP0X

                      *         VERY DOWN ROUTINE

1734: A212    PLAYDWN LDX #18
1736: C6E4            DEC PTX
1738: A014    DWNLOOP3 LDY #20
173A: B1E4    DWNLOOP LDA (PTX),Y
173C: C8              INY
173D: 91E4            STA (PTX),Y
173F: 88              DEY
1740: 88              DEY
1741: 10F7            BPL DWNLOOP         PLAYER ONE LINE SCROLLED
```

```
1743: A010            LDY #16
1745: A9FF    DWNLOOP2 LDA #255
1747: 85F8             STA SHPOH
1749: C6F8    DWNLOOP1 DEC SHPOH
174B: D0FC             BNE DWNLOOP1
174D: 88               DEY
174E: D0F5             BNE DWNLOOP2      SLOW DOWN SCROLLING
1750: CA               DEX
1751: D0E5             BNE DWNLOOP3      PLAYER NOT YET TOTALLY DOWN
1753: 201016 TRIGWAIT JSR STRTWT1        WAIT TRIGGER
1756: 4CCB10           JMP CLRPLX        RTS VIA CLR

PHYSICAL ENDADDRESS: $AF43

*** NO WARNINGS

COUT            $F6A4                   PRIOR           $026F
CONSOL          $D01F                   HITCLR          $D01E
DLISTL          $0230                   DLISTH          $0231
PFCOL0          $02C4                   PFCOL1          $02C5
PFCOL2          $02C6                   PFCOL3          $02C7     UNUSED
PFCOL4          $02C8                   PCOLR0          $02C0
PCOLR1          $02C1                   SDMCTL          $022F
AUDF1           $D200                   AUDC1           $D201
AUDF2           $D202                   AUDC2           $D203
HPOSM0          $D004                   HPOSM1          $D005
HPOSP0          $D000                   HPOSP1          $D001
M0PF            $D000                   M0PL            $D008
M1PF            $D001                   M1PL            $D009
P0PF            $D004                   P0PL            $D00C     UNUSED
P1PF            $D005                   P1PL            $D00D     UNUSED
STICK0          $0278                   STICK1          $0279
STRIG0          $0284                   STRIG1          $0285
DEC1            $E5                     WINFL           $D1
BULLC0          $D2                     BULLC1          $D3
LIFEC0          $D4                     LIFEC1          $D5
SPMBASE         $D6                     LASTFL          $D7
SWTCHFL         $D8                     LEGXFL          $D9
LEG0FL          $DA                     LEG1FL          $DB
LEGC0           $DC                     LEGC1           $DD
SCOUNT0         $DE                     SCOUNT1         $DF
PT0             $E0                     PT1             $E2
PTX             $E4                     MPT0            $E6
MPT1            $E8                     M0COUNT         $EA
M1COUNT         $EB                     SOND0FL         $ED
SOND1FL         $EE                     M0X             $EF
M1X             $F0                     COUNT0          $F1
COUNT1          $F3                     JUMP            $F5
SHP0H           $F8                     SHP1H           $F9
M0FLG           $FA                     M1FLG           $FB
M0VM0FL         $FC                     M0VM1FL         $FD
SAUDF1          $FE                     SAUDF2          $FF
MEMSIZE         $6A                     PMBASE          $D407
MASK0           $03                     MASK01          $FC
MASK1           $0C                     MASK11          $F3
START           $1016     UNUSED        LOOP2           $1032
SETUP0          $104D                   OTHER1          $105B
SETUP1          $105E                   OTHER2          $106C
SETUPX1         $106F                   LOOP3           $1073
LEG2            $107C                   STORE           $107F
```

LOOP31	$1085	SETUPX2	$108E
LOOP031	$1092	LEG21	$109B
STORE01	$109E	LOOP311	$10A4
MOVP1X	$10AD	MOVP0X	$10B6
CLRPL1	$10BF	CLRPL0	$10C5
CLRPLX	$10CB	LOOP4	$10CF
BODYTAB1	$10D5	LEGTAB1	$10E3
LEGTAB2	$10EA	BODYTAB2	$10F1
LEGTAB21	$10FF	LEGTAB22	$1106
MOVP0	$110D	RTS1	$1146
MOVP1	$1147	RTS2	$1180
JMPTAB0	$1181	JMPTAB1	$1191
UP0	$11A1	NOMOV1	$11AE
DOWN0	$11AF	NOMOV2	$11BC
RIGHT0	$11BD	RGTORTS	$11C9
LEFT0	$11CA	LFTORTS	$11D6
UR0	$11D7	UL0	$11DD
DR0	$11E3	DL0	$11E9
UP1	$11EF	NOMOV3	$11FC
DOWN1	$11FD	NOMOV4	$120A
RIGHT1	$120B	RGT1RTS	$1217
LEFT1	$1218	LFT1RTS	$1224
UR1	$1225	UL1	$122B
DR1	$1231	DL1	$1237
COMPARE	$123D	LOOP5	$123F
FOUND	$1249	COMTAB	$124E
CHCKTRIG	$1256	CTRIG1	$1263
CTRGRTS	$126E	MOVM0	$126F
STORE1	$128F	MOVM01	$129C
DECR	$12BB	STORE2	$12BD
RTS3	$12C2	RESM0	$12C3
MOVM1	$12D7	STORE3	$12F7
MOVM11	$1304	DECR1	$1323
STORE4	$1325	RTS4	$132A
RESM1	$132B	SOUND	$133F
RSOUND0	$1358	RTS7	$1366
NOSOND0	$1367	RSOUND1	$1380
SWITCH	$138F	RTS8	$13AB
INIT	$13AC	LOOP1	$141A
LOOP11	$1444	CHCKCOLL	$149C
NOEND0	$14AF	CHCKCOL1	$14CF
NOEND1	$14E2	DISP	$1502
RTS9	$1527	SHOW	$152B
RESTORE	$15CC	STRTWAIT	$1600
STRTWT1	$1610	WAIT10	$1613
WAIT01	$161B	PRINT	$1624
LOOP23	$162C	DECOUT	$1646
DECOUT0	$1648	DECOUT1	$164A
DECOUT3	$1657	DECTAB	$1669
COUT0	$166B	GAMEOVER	$1674
PLONE	$16AE	NUMOUT	$16B0
SPACE	$16C8	PLAYF	$16DD
PLAYDOWN	$16F7	DRUNK1	$170F
DRUNK	$1717	DRNKLOOP	$171D
NODRUNK	$1727	PLAYDWN1	$1731
PLAYDWN	$1734	DWNLOOP3	$1738
DWNLOOP	$173A	DWNLOOP2	$1745
DWNLOOP1	$1749	TRIGWAIT	$1753

```
1000  00 40 00 10 15 10 A9 3C
1008  8D 02 D3 A9 16 85 0A A9
1010  10 85 0B 18 60 60 ÎA2 FF
1018  9A A9 4C 85 F5 A5 6A E9
1020  10 85 D6 A9 08 8D 6F 02
1028  E8 8E C6 02 8E C8 02 20
1030  AC 13 20 0D 11 20 47 11
1038  20 6F 12 20 D7 12 20 56
1040  12 20 9C 14 20 3F 13 20
1048  8F 13 4C 32 10 20 B6 10
1050  A5 DA 85 D9 24 D8 10 03
1058  4C 6F 10 4C 8E 10 20 AD
1060  10 A5 DB 85 D9 24 D8 30
1068  03 4C 6F 10 4C 8E 10 A0
1070  14 A2 06 24 D9 10 05 BD
1078  E3 10 D0 03 BD EA 10 91
1080  E4 88 CA 10 EE B9 D5 10
1088  91 E4 88 10 F8 60 A0 14
1090  A2 06 24 D9 10 05 BD FF
1098  10 D0 03 BD 06 11 91 E4
10A0  88 CA 10 EE B9 F1 10 91
10A8  E4 88 10 F8 60 A5 E2 85
10B0  E4 A5 E3 85 E5 60 A5 E0
10B8  85 E4 A5 E1 85 E5 60 20
10C0  AD 10 4C CB 10 20 B6 10
10C8  4C CB 10 A0 14 A9 00 91
10D0  E4 88 10 FB 60 18 18 7E
10D8  18 38 18 08 1C 1C 1C 7C
10E0  3C 1C 1C 08 08 08 08 08
10E8  38 38 08 08 16 16 12 72
10F0  72 18 18 7E 18 1C 18 10
10F8  38 38 38 3E 3C 38 38 10
1100  10 10 10 10 1C 1C 10 10
1108  68 68 48 4E 4E E6 F1 A5
1110  F1 C9 30 D0 31 A9 00 85
1118  F1 AD 78 02 20 3D 12 B0
1120  25 B9 B1 11 85 F6 B9 82
1128  11 85 F7 20 F5 00 20 4D
1130  10 E6 DC A5 DC C9 08 D0
1138  0D A2 00 8E 1F D0 86 DC
1140  CA 8A 45 DA 85 DA 60 E6
1148  F3 A5 F3 C9 30 D0 31 A9
1150  00 85 F3 AD 79 02 20 3D
1158  12 B0 25 B9 91 11 85 F6
1160  B9 92 11 85 F7 20 F5 00
1168  20 5E 10 E6 DD A5 DD C9
1170  08 D0 0D A2 00 8E 1F D0
1178  86 DD CA 8A 45 DB 85 DB
1180  60 A1 11 AF 11 BD 11 CA
1188  11 D7 11 DD 11 E3 11 E9
1190  11 EF 11 FD 11 0B 12 18
1198  12 25 12 2B 12 31 12 37
11A0  12 20 C5 10 A5 E0 C9 0B
11A8  F0 04 C6 E0 C6 E0 60 20
11B0  C5 10 A5 E0 C9 E3 F0 04
11B8  E6 E0 E6 E0 60 A4 F8 C0
11C0  C9 B0 06 C8 8C 00 D0 84
11C8  F8 60 A4 F8 C0 32 90 06

11D0  88 8C 00 D0 84 F8 60 20
11D8  BD 11 4C A1 11 20 CA 11
11E0  4C A1 11 20 BD 11 4C AF
11E8  11 20 CA 11 4C AF 11 20
11F0  BF 10 A5 E2 C9 0B F0 04
11F8  C6 E2 C6 E2 60 20 BF 10
1200  A5 E2 C9 E3 F0 04 E6 E2
1208  E6 E2 60 A4 F9 C0 C9 B0
1210  06 C8 8C 01 D0 84 F9 60
1218  A4 F9 C0 32 90 06 88 8C
1220  01 D0 84 F9 60 20 0B 12
1228  4C EF 11 20 18 12 4C EF
1230  11 20 0B 12 4C FD 11 20
1238  18 12 4C FD 11 A0 07 D9
1240  4E 12 F0 05 88 10 F8 38
1248  60 98 0A A8 18 60 0E 0D
1250  07 0B 06 0A 05 09 38 AD
1258  84 02 D0 07 A5 D2 F0 03
1260  66 FA 38 AD 85 02 D0 06
1268  A5 D3 F0 02 66 FB 60 24
1270  FC 30 29 A5 FA 85 FC 85
1278  ED 10 47 C6 D2 A9 AF 8D
1280  01 D2 A5 E0 85 E6 A5 F8
1288  24 D8 30 03 18 69 08 85
1290  EF 8D 04 D0 A0 0A B1 E6
1298  09 03 91 E6 E6 EA A5 EA
12A0  C9 10 D0 1E A9 00 85 EA
12A8  A5 EF C9 D0 B0 15 C9 2D
12B0  90 11 24 D8 30 05 E6 EF
12B8  4C BD 12 C6 EF A5 EF 8D
12C0  04 D0 60 46 FC 46 FA 20
12C8  58 13 A0 0A B1 E6 29 FC
12D0  91 E6 A9 00 85 EA 60 24
12D8  FD 30 29 A5 FB 85 FD 85
12E0  EE 10 47 C6 D3 A9 AF 8D
12E8  03 D2 A5 E2 85 E8 A5 F9
12F0  24 D8 10 03 18 69 08 85
12F8  F0 8D 05 D0 A0 0A B1 E8
1300  09 0C 91 E8 E6 EB A5 EB
1308  C9 10 D0 1E A9 00 85 EB
1310  A5 F0 C9 D0 B0 15 C9 2D
1318  90 11 24 D8 10 05 E6 F0
1320  4C 25 13 C6 F0 A5 F0 8D
1328  05 D0 60 46 FD 46 FB 20
1330  80 13 A0 0A B1 E8 29 F3
1338  91 E8 A9 00 85 EB 60 24
1340  ED 10 24 E6 DE A5 DE C9
1348  10 D0 1C A9 00 85 DE E6
1350  FE A5 FE 8D 00 D2 10 0F
1358  A9 A0 8D 01 D2 A9 00 85
1360  FE 8D 00 D2 46 ED 60 24
1368  EE 10 FB E6 DF A5 DF C9
1370  10 D0 F3 A9 00 85 DF E6
1378  FF A5 FF 8D 02 D2 10 E6
1380  A9 A0 8D 03 D2 A9 00 85
1388  FF 8D 02 D2 46 EE 60 A5
1390  F8 C5 F9 66 D8 A5 D8 45
1398  D7 10 10 A5 D8 85 D7 20
```

```
13A0    5E 10 20 4D 10 20 C3 12
13A8    20 2B 13 60 20 28 16 7D
13B0    43 4F 50 59 52 49 47 48
13B8    54 20 31 39 38 31 20 49
13C0    4E 47 2E 20 57 2E 20 48
13C8    4F 46 41 43 4B 45 52 20
13D0    47 4D 42 48 0D 0D 0D 0D
13D8    0D 0D 8D 20 CF 16 20 28
13E0    16 47 55 4E 46 49 47 48
13E8    54 0D 8D 20 CF 16 20 28
13F0    16 42 59 20 48 2E 43 2E
13F8    20 57 41 47 4E 45 52 0D
1400    8D 20 04 16 A9 7D 20 A4
1408    F6 20 E4 16 A9 3B 8D C0
1410    02 A9 8A 8D C1 02 A2 08
1418    A9 00 95 E4 CA 10 FB A9
1420    0B 85 E2 A9 E3 85 E0 A5
1428    D6 18 69 03 85 E7 85 E9
1430    85 E5 18 69 01 85 E1 18
1438    69 01 85 E3 18 69 03 85
1440    D7 A0 00 98 91 E4 C8 D0
1448    FB E6 E5 A6 E5 E4 D7 90
1450    F3 A9 37 8D 00 D0 85 F8
1458    A9 C8 8D 01 D0 85 F9 46
1460    FB 46 FA A9 00 85 DC 85
1468    DD 85 D8 85 D7 A9 0A 85
1470    D4 85 D5 A9 32 85 D2 85
1478    D3 A9 CC 8D C5 02 A9 1F
1480    8D C8 02 A9 0D 8D C4 02
1488    A9 8C 8D C6 02 A9 03 8D
1490    1D D0 20 4D 10 20 5E 10
1498    8D 1E D0 60 A9 FF 85 D1
14A0    AD 04 D0 49 FF 2D 00 D0
14A8    29 01 F0 03 20 C3 12 AD
14B0    04 D0 29 04 0D 09 D0 29
14B8    05 F0 14 20 2B 13 20 C3
14C0    12 20 FE 16 A9 32 85 D2
14C8    C6 D4 D0 36 4C 7B 16 A9
14D0    00 85 D1 AD 05 D0 49 FF
14D8    2D 01 D0 29 01 F0 03 20
14E0    2B 13 AD 05 D0 29 04 0D
14E8    08 D0 29 06 F0 39 20 C3
14F0    12 20 2B 13 20 FE 16 A9
14F8    32 85 D3 C6 D5 D0 03 4C
1500    7B 16 20 C5 10 20 BF 10
1508    A9 0B 85 E2 A9 E3 85 E0
1510    A9 37 85 F8 8D 00 D0 A9
1518    C8 85 F9 8D 01 D0 20 4D
1520    10 20 5E 10 20 2B 15 8D
1528    1E D0 60 20 D0 15 20 28
1530    16 0D 50 4C 41 59 45 52
1538    20 31 20 3A 0D 0D 0D 42
1540    55 4C 4C 45 54 53 20 BA
1548    A5 D2 8D E5 00 20 4A 16
1550    20 28 16 0D 43 4F 57 42
1558    4F 59 53 20 BA A5 D4 8D
1560    E5 00 20 4A 16 20 28 16
1568    0D 0D 0D 50 4C 41 59 45
1570    52 20 32 20 3A 0D 0D 0D
1578    42 55 4C 4C 45 54 53 20

1580    BA A5 D3 8D E5 00 20 4A
1588    16 20 28 16 0D 43 4F 57
1590    42 4F 59 53 20 BA A5 D5
1598    8D E5 00 20 4A 16 20 28
15A0    16 0D 8D 20 04 16 A9 7D
15A8    20 A4 F6 20 E4 16 A9 3B
15B0    8D C0 02 A9 8A 8D C1 02
15B8    A9 8C 8D C6 02 A9 1F 8D
15C0    C8 02 A9 CC 8D C5 02 A9
15C8    03 8D 1D D0 8D 1E D0 60
15D0    A9 00 8D 1D D0 8D C6 02
15D8    8D C1 02 8D C0 02 8D C8
15E0    02 A9 0A 8D C5 02 A9 00
15E8    8D 2F 02 A5 59 8D 31 02
15F0    A9 20 8D 30 02 A5 D6 8D
15F8    07 D4 A9 3E 8D 2F 02 A9
1600    7D 4C 72 16 20 CF 16 20
1608    28 16 28 54 52 49 47 47
1610    45 52 29 8D 20 1F 16 AD
1618    84 02 2D 85 02 D0 F8 AD
1620    84 02 2D 85 02 F0 F8 60
1628    68 85 E4 68 85 E5 A2 00
1630    E6 E4 D0 02 E6 E5 A1 E4
1638    29 7F 20 72 16 A2 00 A1
1640    E4 10 ED A5 E5 48 A5 E4
1648    48 60 A0 01 A2 30 38 AD
1650    E5 00 F9 70 16 90 06 8D
1658    E5 00 E8 D0 F1 8A 84 E4
1660    20 72 16 A4 E4 88 10 E4
1668    AD E5 00 09 30 4C 72 16
1670    0A 64 C9 0D D0 02 A9 9B
1678    4C A4 F6 20 58 13 20 80
1680    13 20 D0 15 20 28 16 0D
1688    0D 0D 0D 0D 0D 0D 8D 20
1690    CF 16 20 28 16 47 41 4D
1698    45 20 4F 56 45 52 0D 8D
16A0    20 CF 16 20 28 16 50 4C
16A8    41 59 45 52 A0 24 D1 10
16B0    04 A9 32 D0 02 A9 31 20
16B8    A4 F6 20 28 16 20 57 49
16C0    4E 53 21 21 21 0D 0D 0D
16C8    8D 20 04 16 4C AC 13 20
16D0    28 16 20 20 20 20 20 20
16D8    20 20 20 20 20 20 20 20
16E0    20 20 A0 60 A9 00 8D 2F
16E8    02 A9 98 8D 30 02 A9 1F
16F0    8D 31 02 A5 D6 8D 07 D4
16F8    A9 3E 8D 2F 02 60 AD 04
1700    D0 0D 05 D0 29 04 F0 26
1708    24 D1 30 0A 20 BF 10 A9
1710    72 8D 01 D0 D0 08 20 C5
1718    10 A9 72 8D 00 D0 A9 80
1720    85 E4 A0 02 B9 D5 10 91
1728    E4 88 10 F8 30 2C 24 D1
1730    30 06 20 AD 10 4C 3B 17
1738    20 B6 10 A2 12 C6 E4 A0
1740    14 B1 E4 C8 91 E4 88 88
1748    10 F7 A0 0C A9 FF 85 F8
1750    C6 F8 D0 FC 88 D0 F5 CA
1758    D0 E5 20 14 16 4C CB 10
```

```
1F98  70 70 70 4D 60 20 0D 0D
1FA0  0D 0D 0D 0D 0D 0D 0D 0D
1FA8  0D 0D 0D 0D 0D 0D 0D 0D
1FB0  0D 0D 0D 0D 0D 0D 0D 0D
1FB8  0D 0D 0D 0D 0D 0D 0D 0D
1FC0  0D 0D 0D 0D 0D 0D 0D 0D
1FC8  0D 0D 0D 0D 0D 0D 0D 0D
1FD0  0D 0D 0D 0D 0D 0D 0D 0D
1FD8  0D 0D 0D 0D 0D 0D 0D 0D
1FE0  0D 0D 0D 0D 0D 0D 0D 0D
1FE8  0D 0D 0D 0D 0D 0D 0D 0D
1FF0  0D 0D 0D 0D 0D 0D 0D 0D
1FF8  0D 0D 0D 0D 0D 41 98 1F
2000  00 00 00 00 00 00 00 00
2008  00 00 00 00 00 00 00 00
2010  00 00 00 00 00 00 00 00
2018  00 00 00 00 00 00 00 00
2020  00 00 00 00 00 00 00 00
2028  00 00 00 00 00 00 00 00
2030  00 00 00 00 00 00 00 00
2038  00 00 00 00 00 00 00 00
2040  00 00 00 00 00 00 00 00
2048  00 00 00 00 00 00 00 00
2050  00 00 00 00 00 00 00 00
2058  00 00 00 00 00 00 00 00
2060  00 00 00 00 00 00 00 00
2068  00 00 00 00 00 00 00 00
2070  00 00 00 00 00 00 00 00
2078  00 00 00 00 00 00 00 00
2080  00 00 00 00 00 00 00 00
2088  00 00 00 00 00 00 00 00
2090  00 00 00 00 00 00 00 00
2098  00 00 00 00 00 00 00 00
20A0  00 00 00 00 00 00 00 00
20A8  00 00 00 00 00 00 00 00
20B0  00 00 00 00 00 00 00 00
20B8  00 00 00 00 00 00 00 00
20C0  00 00 00 00 00 00 00 00
20C8  00 00 00 00 00 00 00 00
20D0  00 00 00 00 00 00 00 00
20D8  00 00 00 00 00 00 00 00
20E0  00 00 00 00 00 00 00 00
20E8  00 00 00 00 00 00 00 00
20F0  00 00 00 00 00 00 00 00
20F8  00 00 00 00 00 00 00 00
2100  00 00 00 00 00 00 00 00
2108  00 00 00 00 00 00 00 00
2110  00 00 00 00 00 00 00 00
2118  00 00 00 00 00 00 00 00
2120  00 00 00 00 00 00 00 00
2128  00 00 00 00 00 00 00 00
2130  00 00 00 00 00 00 00 00
2138  00 00 00 00 00 00 00 00
2140  00 00 00 00 00 00 00 00
2148  00 00 00 00 00 00 00 00
2150  00 00 00 00 00 00 00 00
2158  00 00 00 00 00 00 00 00
2160  00 00 00 00 00 00 00 00
```

```
2168  00 00 00 00 00 00 00 00
2170  00 00 00 00 00 00 00 00
2178  00 00 00 00 00 00 00 00
2180  00 00 00 00 00 00 00 00
2188  00 00 00 00 00 00 00 00
2190  00 00 00 00 00 00 00 00
2198  00 00 00 00 00 00 00 00
21A0  00 00 00 00 00 00 00 00
21A8  00 00 00 00 00 00 00 00
21B0  00 00 00 00 00 00 00 00
21B8  00 00 00 00 00 00 00 00
21C0  00 00 00 00 00 00 00 00
21C8  00 00 00 00 00 00 00 00
21D0  00 00 00 00 00 00 00 00
21D8  00 00 00 00 00 00 00 00
21E0  00 00 00 00 00 00 00 00
21E8  54 00 00 00 00 00 00 00
21F0  00 00 00 00 00 00 00 0A
21F8  02 80 00 00 00 00 00 00
2200  00 00 00 00 00 00 00 00
2208  00 00 00 00 00 00 00 01
2210  55 00 00 00 00 00 00 00
2218  00 00 00 00 00 00 00 0A
2220  02 80 00 00 00 00 00 00
2228  00 00 00 00 00 00 00 00
2230  00 00 00 00 00 00 00 01
2238  55 55 00 00 00 00 00 00
2240  00 00 00 00 00 00 00 0A
2248  02 80 00 00 00 00 00 00
2250  00 00 00 00 00 00 00 00
2258  00 00 00 00 00 00 00 05
2260  55 55 50 00 00 00 00 00
2268  00 00 00 00 00 00 00 0A
2270  02 80 00 00 00 00 00 00
2278  00 00 00 00 00 00 00 00
2280  00 00 00 00 00 00 00 05
2288  55 55 54 00 00 00 00 00
2290  00 00 00 00 00 00 00 0A
2298  AA 80 00 00 00 00 00 00
22A0  00 00 00 00 00 00 00 00
22A8  00 00 00 00 00 00 00 05
22B0  55 55 54 00 00 00 00 00
22B8  00 00 00 00 00 00 00 0A
22C0  AA 80 00 00 00 00 00 00
22C8  00 00 00 00 00 00 00 00
22D0  00 00 00 00 00 00 00 15
22D8  55 55 55 40 00 00 00 00
22E0  00 00 00 00 00 00 00 00
22E8  02 80 00 00 00 00 00 00
22F0  00 00 00 00 00 00 00 00
22F8  00 00 00 00 00 00 00 15
2300  55 55 55 50 00 00 00 00
2308  00 00 00 00 00 00 00 00
2310  02 80 28 00 00 00 00 00
2318  00 00 00 00 00 00 00 00
2320  00 00 00 00 00 00 00 55
2328  55 55 55 55 00 00 00 00
2330  00 00 00 00 00 00 00 00
2338  02 80 28 00 00 00 00 00
```

```
2340  00 00 00 00 00 00 00 00
2348  00 00 00 00 00 00 00 55
2350  55 55 55 55 50 00 00 00
2358  00 00 00 00 00 00 00 00
2360  02 80 28 00 00 00 00 00
2368  00 00 00 00 00 00 00 00
2370  00 00 00 00 00 00 00 55
2378  55 55 55 55 54 00 00 00
2380  00 00 00 00 00 00 00 00
2388  02 80 28 00 00 00 00 00
2390  00 00 00 00 00 00 00 00
2398  00 00 00 00 00 00 01 55
23A0  55 55 55 55 55 50 00 00
23A8  00 00 00 00 00 00 00 00
23B0  02 AA A8 00 00 00 00 00
23B8  00 00 00 00 00 00 0F FF
23C0  FF FF 00 00 00 00 05 55
23C8  55 55 55 55 55 55 00 00
23D0  00 00 00 00 00 00 00 00
23D8  02 AA A8 00 00 00 00 00
23E0  00 00 00 00 00 3F FF FF
23E8  FF FF FC 00 00 00 05 55
23F0  55 55 55 55 55 55 54 00
23F8  00 00 00 00 00 00 00 00
2400  02 80 00 00 00 00 00 00
2408  00 00 00 00 0F FF FF FF
2410  FF FF FF 00 00 00 05 55
2418  55 55 55 55 55 55 55 40
2420  00 00 00 00 00 00 00 00
2428  02 80 00 00 00 00 00 00
2430  00 00 00 00 FF FF FF FF
2438  FF FF FF C0 00 00 55 55
2440  55 55 55 55 55 55 55 55
2448  00 00 00 00 00 00 00 00
2450  02 80 00 00 00 00 00 00
2458  00 00 00 03 FF FF FF FF
2460  FF FF FF C0 00 00 00 00
2468  00 00 00 00 00 00 00 00
2470  00 00 00 00 00 00 00 00
2478  00 00 00 00 00 00 00 00
2480  00 00 00 0F FF FF FF FF
2488  FF FF FF 00 00 00 00 00
2490  00 00 00 00 00 00 00 00
2498  00 00 00 00 00 00 00 00
24A0  00 00 00 00 00 00 00 00
24A8  00 00 00 3F FF FF FF FF
24B0  FF FF F0 00 00 00 00 00
24B8  00 00 00 00 00 00 00 00
24C0  00 00 00 00 00 00 00 00
24C8  00 00 00 00 00 00 00 00
24D0  00 00 03 FF FF FF FF FF
24D8  FF FF C0 00 00 00 00 00
24E0  00 00 00 00 00 00 00 00
24E8  00 00 00 00 00 00 00 00
24F0  00 00 00 00 00 00 00 00
24F8  00 00 03 FF FF FF FF FF
2500  FF F0 00 00 00 00 00 00
2508  00 00 00 00 00 00 00 00

2510  00 00 00 00 00 00 00 00
2518  00 00 00 00 00 00 00 00
2520  00 00 0F FF FF FF FF FF
2528  FF 00 00 00 00 00 00 00
2530  00 00 00 00 00 00 00 00
2538  00 00 00 00 00 00 00 00
2540  00 00 00 00 00 00 00 00
2548  00 00 3F FF FF FF FF FF
2550  FC 00 00 00 00 00 00 00
2558  00 00 00 00 00 00 00 00
2560  00 00 00 00 00 00 00 00
2568  00 00 00 00 00 00 00 00
2570  00 03 FF FF FF FF FF FF
2578  C0 00 00 00 00 00 00 00
2580  00 00 00 00 00 00 00 00
2588  00 00 00 00 00 00 00 00
2590  00 00 00 00 00 00 00 00
2598  00 03 FF FF FF FF FF FC
25A0  00 00 00 00 00 00 00 00
25A8  00 00 00 00 00 00 00 00
25B0  00 00 00 00 00 00 00 00
25B8  00 00 00 00 00 00 00 00
25C0  00 0F FF FF FF FF FF C0
25C8  00 00 00 00 00 00 00 00
25D0  00 00 00 00 00 00 00 00
25D8  00 00 00 00 00 00 00 00
25E0  00 00 00 00 00 00 00 00
25E8  00 0F FF FF FF FF FF 00
25F0  00 00 00 00 00 00 00 00
25F8  00 00 00 00 00 00 00 00
2600  00 00 00 00 00 00 00 00
2608  00 00 00 00 00 00 00 00
2610  00 0F FF FF FF FF FF 00
2618  00 00 00 00 00 00 00 00
2620  00 00 00 00 00 00 00 00
2628  00 00 00 00 00 00 00 00
2630  00 00 00 00 00 00 00 00
2638  00 3F FF FF FF FF FC 00
2640  00 00 00 00 00 00 00 00
2648  00 00 00 00 00 00 00 00
2650  00 00 00 00 00 00 00 00
2658  00 00 00 00 00 00 00 00
2660  00 FF FF FF FF FF F0 00
2668  00 00 00 00 00 00 00 00
2670  00 00 00 00 00 00 00 00
2678  00 00 00 00 00 00 00 00
2680  00 00 00 00 00 00 00 00
2688  0F FF FF FF FF FF C0 00
2690  00 00 00 00 00 00 00 00
2698  00 00 00 00 00 00 00 00
26A0  00 00 28 00 00 00 00 00
26A8  00 00 00 00 00 00 00 00
26B0  FF FF FF FF FF FF 00 00
26B8  00 00 00 00 00 00 00 00
26C0  0A 02 80 00 00 00 00 00
26C8  00 00 28 00 00 00 00 00
26D0  00 00 00 00 00 00 00 3F
26D8  FF FF FF FF FF F0 00 00
```

```
26E0  00 00 00 00 00 00 00 00
26E8  0A 02 80 00 00 00 00 00
26F0  00 00 28 00 00 00 00 00
26F8  00 00 00 00 00 00 0F FF
2700  FF FF FF FF FF C0 00 00
2708  00 00 00 00 00 00 00 00
2710  0A 02 80 00 00 00 00 00
2718  00 00 28 0A 00 00 00 00
2720  00 00 00 00 00 00 FF FF
2728  FF FF FF FF FF C0 00 00
2730  00 00 00 00 00 00 00 00
2738  0A 02 80 00 00 00 00 00
2740  00 00 28 0A 00 00 00 00
2748  00 00 00 00 00 3F FF FF
2750  FF FF FF FF FF F0 00 00
2758  00 00 00 00 00 00 00 00
2760  0A AA 80 00 00 00 00 00
2768  00 00 2A AA 00 00 00 00
2770  00 00 00 00 0F FF FF FF
2778  FF FF FF FF FF F0 00 00
2780  00 00 00 00 00 00 00 00
2788  0A AA 80 00 00 00 00 00
2790  00 00 2A AA 00 00 00 00
2798  00 00 00 00 3F FF FF FF
27A0  FF FF FF FF FF FF 00 00
27A8  00 00 00 00 00 00 00 00
27B0  00 02 80 00 00 00 00 00
27B8  00 00 00 0A 00 00 00 00
27C0  00 00 00 03 FF FF FF FF
27C8  FF FF FF FF FF FF F0 00
27D0  00 00 00 00 00 00 00 00
27D8  00 02 80 28 00 00 00 00
27E0  00 00 00 0A 00 00 00 00
27E8  00 00 00 FF FF FF FF FF
27F0  FF FF FF FF FF FF FF 00
27F8  00 00 00 00 00 00 00 00
2800  00 02 80 28 00 00 00 00
2808  00 00 00 0A 02 80 00 00
2810  00 00 03 FF FF FF FF FF
2818  FF FF FF FF FF FF FF FF
2820  FF FC 00 00 00 00 00 00
2828  00 02 80 28 00 00 00 00
2830  00 00 00 0A 02 80 00 00
2838  00 00 00 FF FF FF FF FF
2840  FF FF FF FF FF FF FF FF
2848  FF FF FF FF 00 00 00 00
2850  00 02 80 28 00 00 00 00
2858  00 00 00 0A AA 80 00 00
2860  00 00 00 3F FF FF FF FF
2868  FF FF FF FF FF FF FF FF
2870  FF FF FF FF F0 00 00 00
2878  00 02 AA A8 00 00 00 00
2880  00 00 00 0A AA 80 00 00
2888  00 00 00 0F FF FF FF FF
2890  FF FF FF FF FF FF FF FF
2898  FF FF FF FF FC 00 00 00
28A0  00 02 AA A8 00 00 00 00
28A8  00 00 00 0A 00 00 00 00

28B0  00 00 00 03 FF FF FF FF
28B8  FF FF FF FF FF FF FF FF
28C0  FF FF FF FF C0 00 00 00
28C8  00 02 80 00 00 00 00 00
28D0  00 00 00 0A 00 00 00 00
28D8  00 00 00 00 FF FF FF FF
28E0  FF FF FF FF FF FF FF FF
28E8  FF FF FF FC 00 00 00 00
28F0  00 02 80 00 00 00 00 00
28F8  00 00 00 0A 00 00 00 00
2900  00 00 00 00 0F FF FF FF
2908  FF FF FF FF FF FF FC 00
2910  00 00 00 00 00 00 00 00
2918  00 02 80 00 00 00 00 00
2920  00 00 00 0A 00 00 00 00
2928  00 00 00 00 00 00 00 FF
2930  FF FF FF FF C0 00 00 00
2938  00 00 00 00 00 00 00 00
2940  00 00 00 00 00 00 00 00
2948  00 00 00 00 00 00 00 00
2950  00 00 00 00 00 00 00 00
2958  00 00 00 00 00 00 00 00
2960  00 00 00 00 00 00 00 00
2968  00 00 00 00 00 00 00 00
2970  00 00 00 00 00 00 00 00
2978  00 00 00 00 00 00 00 00
2980  00 00 00 00 00 00 00 00
2988  00 00 00 00 00 00 00 00
2990  00 00 00 00 00 00 00 00
2998  00 00 00 00 00 00 00 00
29A0  00 00 00 00 00 00 00 00
29A8  00 00 00 00 00 00 00 00
29B0  00 00 00 00 00 00 00 00
29B8  00 00 00 00 00 00 00 00
29C0  00 00 00 00 00 00 00 00
29C8  00 00 00 00 00 00 00 00
29D0  00 00 00 00 00 00 00 00
29D8  00 00 00 00 00 00 00 00
29E0  00 00 00 00 00 00 00 00
29E8  00 00 00 00 00 00 00 00
29F0  00 00 00 00 00 00 00 00
29F8  00 00 00 00 00 00 00 00
2A00  00 00 00 00 00 00 00 00
2A08  00 00 00 00 00 00 00 00
2A10  00 00 00 00 00 00 00 00
2A18  00 00 00 00 00 00 00 00
2A20  00 00 00 00 00 00 00 00
2A28  00 00 00 00 00 00 00 00
2A30  00 00 00 00 00 00 00 00
2A38  00 00 00 00 00 00 00 00
2A40  00 00 00 00 00 00 00 00
2A48  00 00 00 00 00 00 00 00
2A50  00 00 00 00 00 00 00 00
2A58  00 00 00 00 00 00 00 00
2A60  00 00 00 00 00 00 00 00
2A68  00 00 00 00 00 00 00 00
2A70  00 00 00 00 00 00 00 00
2A78  00 00 00 00 00 00 00 00
```

```
2A80  00 00 00 00 00 00 00 00
2A88  00 00 00 00 00 00 00 00
2A90  00 00 00 00 00 00 00 00
2A98  00 00 00 00 00 00 00 00
2AA0  00 00 00 00 00 00 00 00
2AA8  00 00 00 00 00 00 00 00
2AB0  00 00 00 00 00 00 00 00
2AB8  00 00 00 00 00 00 00 00
2AC0  00 00 00 00 00 00 00 00
2AC8  00 00 00 00 00 00 00 00
2AD0  00 00 00 00 00 00 00 00
2AD8  00 00 00 00 00 00 00 00
2AE0  00 00 00 00 00 00 00 00
2AE8  00 00 00 00 00 00 00 00
2AF0  00 00 00 00 00 00 00 00
2AF8  00 00 00 00 00 00 00 00
2B00  00 00 00 00 00 00 00 00
2B08  00 00 00 00 00 00 00 00
2B10  00 00 00 00 00 00 00 00
2B18  00 00 00 00 00 00 00 00
2B20  00 00 00 00 00 00 00 00
2B28  00 00 00 00 00 00 00 00
2B30  00 00 00 00 00 00 00 00
2B38  00 00 00 00 00 00 00 00
2B40  00 00 00 00 00 00 00 00
2B48  00 00 00 00 00 00 00 00
2B50  00 00 00 00 00 00 00 00
2B58  00 00 00 00 00 00 00 00
2B60  00 00 00 00 A0 28 00 00
2B68  00 00 00 00 00 00 55 00
2B70  00 00 00 00 00 00 00 00
2B78  00 00 00 00 00 00 00 00
2B80  00 00 00 00 00 00 00 00
2B88  00 00 00 00 A0 28 00 00
2B90  00 00 00 00 00 05 55 54
2B98  00 00 00 00 00 00 00 00
2BA0  00 00 00 00 00 00 00 00
2BA8  00 00 00 00 00 00 00 00
2BB0  00 00 00 00 A0 28 00 00
2BB8  00 00 00 00 00 05 55 55
2BC0  40 00 00 00 00 00 00 00
2BC8  00 00 00 00 00 00 00 00
2BD0  00 00 00 00 00 00 00 00
2BD8  00 00 00 00 A0 28 00 00
2BE0  00 00 00 00 00 05 55 55
2BE8  55 00 00 00 00 00 00 00
2BF0  00 00 00 00 00 00 00 00
2BF8  00 00 00 00 00 00 00 00
2C00  00 00 00 00 AA A8 00 00
2C08  00 00 00 00 00 15 55 55
2C10  55 40 00 00 00 00 00 00
2C18  00 00 00 00 00 00 00 00
2C20  00 00 00 00 00 00 00 00
2C28  00 00 00 00 AA A8 00 00
2C30  00 00 00 00 00 15 55 55
2C38  55 50 00 00 00 00 00 00
2C40  00 00 00 00 00 00 00 00
2C48  00 00 00 00 00 00 00 00

2C50  00 00 00 00 00 28 00 00
2C58  00 00 00 00 00 55 55 55
2C60  55 55 00 00 00 00 00 00
2C68  00 00 00 00 00 00 00 00
2C70  15 40 00 00 00 00 00 00
2C78  00 00 00 00 00 28 02 80
2C80  00 00 00 00 01 55 55 55
2C88  55 55 50 00 00 00 00 00
2C90  00 00 00 00 00 00 00 05
2C98  55 54 00 00 00 00 00 00
2CA0  00 00 00 00 00 28 02 80
2CA8  00 00 00 00 01 55 55 55
2CB0  55 55 54 00 00 00 00 00
2CB8  00 00 00 00 00 00 00 15
2CC0  55 55 00 00 00 00 00 00
2CC8  00 00 00 00 00 28 02 80
2CD0  00 00 00 00 05 55 55 55
2CD8  55 55 55 00 00 00 00 00
2CE0  00 00 00 00 00 00 00 55
2CE8  55 55 00 00 00 00 00 00
2CF0  00 00 00 00 00 28 02 80
2CF8  00 00 00 00 05 55 55 55
2D00  55 55 55 00 00 00 00 00
2D08  00 00 00 00 00 00 00 55
2D10  55 55 00 00 00 00 00 00
2D18  00 00 00 00 00 2A AA 80
2D20  00 00 00 00 05 55 55 55
2D28  55 55 55 50 00 00 00 00
2D30  00 00 00 00 00 00 01 55
2D38  55 55 00 00 00 00 00 00
2D40  00 00 00 00 00 2A AA 80
2D48  00 00 00 00 55 55 55 55
2D50  55 55 55 54 00 00 00 00
2D58  00 00 00 00 00 00 01 55
2D60  55 55 00 00 00 00 00 00
2D68  00 00 00 00 00 28 00 00
2D70  00 00 00 00 55 55 55 55
2D78  55 55 55 55 50 00 00 00
2D80  00 00 00 00 00 00 15 55
2D88  55 55 40 00 00 00 00 00
2D90  00 00 00 00 00 28 00 00
2D98  00 00 00 01 55 55 55 55
2DA0  55 55 55 55 54 00 00 00
2DA8  00 00 00 00 00 00 55 55
2DB0  55 55 40 00 00 00 00 00
2DB8  00 00 00 00 00 28 00 00
2DC0  00 00 00 00 00 00 00 00
2DC8  00 00 00 00 00 00 00 00
2DD0  00 00 00 00 00 05 55 55
2DD8  55 55 40 00 00 00 00 00
2DE0  00 00 00 00 00 00 00 00
2DE8  00 00 00 00 00 00 00 00
2DF0  00 00 00 00 00 00 00 00
2DF8  00 00 00 00 00 15 55 55
2E00  55 55 50 00 00 00 00 00
2E08  00 00 00 00 00 00 00 00
2E10  00 00 00 00 00 00 00 00
2E18  00 00 00 00 00 00 00 00
```

```
2E20    00 00 00 00 00 15 55 55
2E28    55 55 50 00 00 00 00 00
2E30    00 00 00 00 00 00 00 00
2E38    00 00 00 00 00 00 00 00
2E40    00 00 00 00 00 00 00 00
2E48    00 00 00 00 01 55 55 55
2E50    55 55 50 00 00 00 00 00
2E58    00 00 00 00 00 00 00 00
2E60    00 00 00 00 00 00 00 00
2E68    00 00 00 00 00 00 00 00
2E70    00 00 00 00 05 55 55 55
2E78    55 55 50 00 00 00 00 00
2E80    00 00 00 00 00 00 00 00
2E88    00 00 00 00 00 00 00 00
2E90    00 00 00 00 00 00 00 00
2E98    00 00 00 00 15 55 55 55
2EA0    55 55 54 00 00 00 00 00
2EA8    00 00 00 00 00 00 00 00
2EB0    00 00 00 00 00 00 00 00
2EB8    00 00 00 00 00 00 00 00
2EC0    00 00 00 00 15 55 55 55
2EC8    55 55 54 00 00 00 00 00
2ED0    00 00 00 00 00 00 00 00
2ED8    00 00 00 00 00 00 00 00
2EE0    00 00 00 00 00 00 00 00
2EE8    00 00 00 00 55 55 55 55
2EF0    55 55 54 00 00 00 00 00
2EF8    00 00 00 00 00 00 00 00
2F00    00 00 00 00 00 00 00 00
2F08    00 00 00 00 00 00 00 00
2F10    00 00 00 05 55 55 55 55
2F18    55 55 54 00 00 00 00 00
2F20    00 00 00 00 00 00 00 00
2F28    00 00 00 00 00 00 00 00
2F30    00 00 00 00 00 00 00 00
2F38    00 00 00 05 55 55 55 55
2F40    55 55 55 40 00 00 00 00
2F48    00 00 00 00 00 00 00 00
2F50    00 00 00 00 00 00 00 00
2F58    00 00 00 00 00 00 00 00
2F60    00 00 00 00 00 00 00 00
2F68    00 00 00 00 00 00 00 00
2F70    00 00 00 00 00 00 00 00
2F78    00 00 00 00 00 00 00 00
2F80    00 00 00 00 00 00 00 00
2F88    00 00 00 00 00 00 00 00
2F90    00 00 00 00 00 00 00 00
2F98    00 00 00 00 00 00 00 00
2FA0    00 00 00 00 00 00 00 00
2FA8    00 00 00 00 00 00 00 00
2FB0    00 00 00 00 00 00 00 00
2FB8    00 00 00 00 00 00 00 00
2FC0    00 00 00 00 00 00 00 00
2FC8    00 00 00 00 00 00 00 00
2FD0    00 00 00 00 00 00 00 00
2FD8    00 00 00 00 00 00 00 00
2FE0    00 00 00 00 00 00 00 00
2FE8    00 00 00 00 00 00 00 00
2FF0    00 00 00 00 00 00 00 00
2FF8    00 00 00 00 00 00 00 00
```

APPENDIX

A. Important addresses

1. Color and luminance of background: 712
   Color: 0–240 in steps of 16 (grey, gold, orange, red-orange, pink, purple, purple-blue, blue, blue, light-blue, turquoise, green-blue, green, yellow-green, orange-green, light-orange)
   luminance: 0 (black) – 15 (white)
   example: POKE 712,16+8 produces a golden background of medium luminance
2. color and luminance of playfield 0 : 708
3. color and luminance of playfield I : 709
4. color and luminance of playfield 2 : 710
5. color and luminance of playfield 3 : 711
6. color and luminance of player-missile 0 : 704
7. color and luminance of player-missile 1 : 705
8. color and luminance of player-missile 2 : 706
9. color and luminance of player-missile 3 : 707
10. graphic control : 53277
    example: POKE 53277,3
    enables player-missile graphics
11. collision clear : 53278
12. horizontal position of missile 0 : 53252
13. horizontal position of missile 1 : 53253
14. horizontal position of missile 2 : 53254
15. horizontal position of missile 3 : 53255
16. horizontal position of player 0 : 53248
17. horizontal position of player 1 : 53249
18. horizontal position of player 2 : 53250
19. horizontal position of player 3 : 53251
20. missile 0 to playfield collisions : 53248
21. missile 0 to player collisions : 53256
22. missile 1 to playfield collisions : 53249
23. missile 1 to player collisions : 53257
24. missile 2 to playfield collisions : 53250
25. missile 2 to player collisions : 53258
26. missile 3 to playfield collisions : 53251
27. missile 3 to player collisions : 53259
28. player 0 to playfield collisions : 53252
29. player 0 to player collisions : 53260

30. player 1 to playfield collisions . 53253
31. player 1 to player collisions : 53261
32. player 2 to playfield collisions : 53254
33. player 2 to player collisions . 53262
34. player 3 to playfield collisions : 53255
35. player 3 to player collisions : 53263
36. player missile base address : 54279
37. priority select : 623

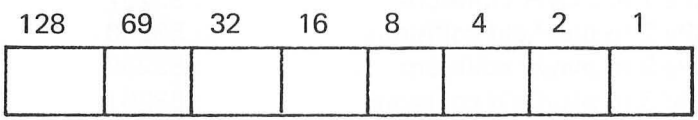
example: POKE 623,1 selects the following ranking:
Player 0, player 1, player 2, player 3, playfield 0, playfield 1, playfield 2, playfield 3, background

or POKE 623,8 selects the following rankings
PF0, PF1, P0, P1, P2, P3, PF2, PF3, BAK

38. random number generator : 53770
39. size for all missiles : 53260
example: POKE 53260,1 makes missile 0 twice normal size
or: POKE 53260,255 makes all four missiles four times normal 1.
40. size of player 0 : 53256
example: POKE 53256,1 twice
or: POKE 53256,3 four times normal size
41. size of player 1: 53257
42. size of player 2: 53258
43. size of player 3: 53259

B. How to create shapes

Each player is eight dots wide. Each dot can be turned on separately by poking different values into the player missile memory. The far left dot has the highest and the far right dot has the lowest value.
Values of the dots:

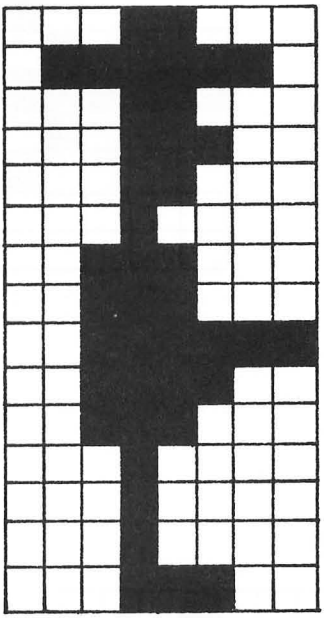


Add the values of all dots you want to turn on together and poke the sum into memory.

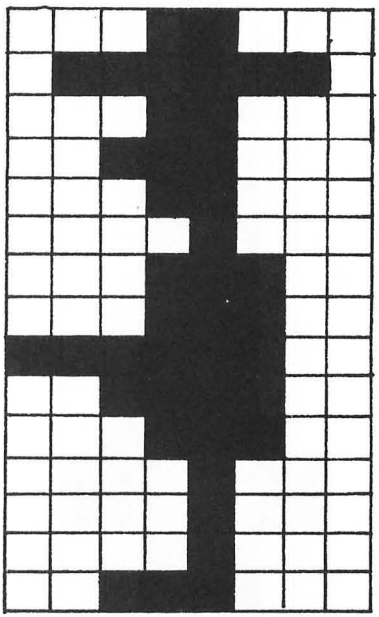
Example: you want to turn on all dots:
128+64+32+16+8+4+2+1 = 255

POKE PMBAS + Y, 255 will create a bar.

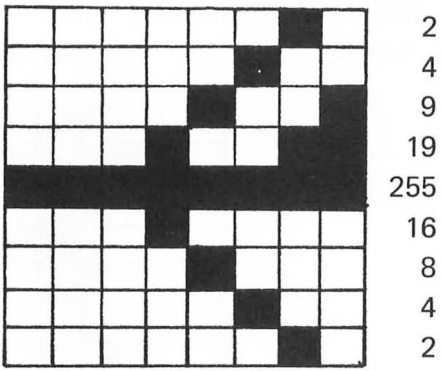
1. cowboy



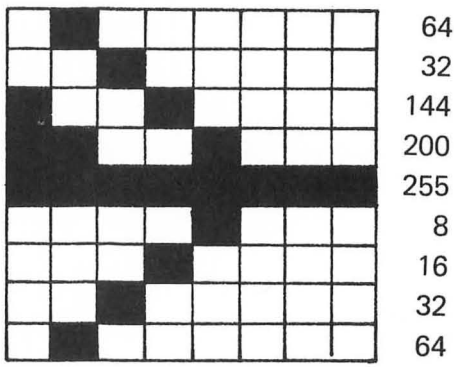
2. **other cowboy**



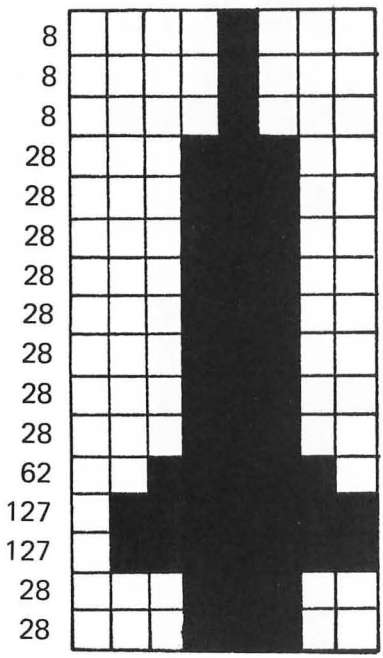
3. bomber



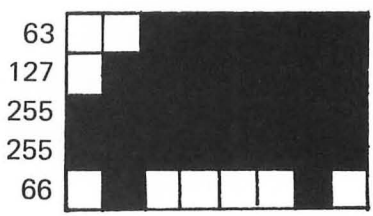
4. other bomber



5. rocket



6. car



best looking sizes: normal size for no. 5
twice normal size for no. 1, 2, 3,4, 6

C. Special effects

I. Explosion

change very quickly color background

example: poke random numbers into color register of background

```
10 POKE 712, PEEK (53770): GOTO 10
```

In addition, sound may be added.

2. earthquake:

move the whole screen a little bit in a horizontal direction

example:

```
10 A=PEEK (560) = 256 * PEEK (561)
10 POKE A+4, 64+PEEK (53770) / 200
30 GOTO 20
```

ANTIC

To use the following programs on ATARIs with less than 48K RAM, please refer to "Important Notice" at the beginning of the book.

The video processor "ANTIC" and the ATARI 400/800 computer
The ATARI 400/800 computer uses a separate microprocessor, called "ANTIC", for generating a picture on the TV screen. This processor uses it's own control program, called the "display list". To understand the interaction between the microprocessor and the TV screen, one must understand how a picture is created on a TV screen. Fig. 1 shows this.

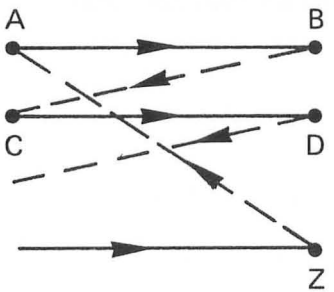


Fig. 1

An electron beam is deflected from point A to point B. As it sweeps across the screen, it's intensity changes, changing the brightness on the screen.
At point B, the beam is darkened and brought back to C. There it is brightened and starts sweeping again, this time to point D. It continues in the same way until it reaches point Z. It is then darkened and returned to point A, and a new picture begins. Between A and Z, the beam sweeps the screen 262 times. Normally, there are 525 lines on a TV screen, but only one half-picture, or 262 lines, is drawn at once.
Most TV sets are overscanned. That is, the picture is larger than the screen. Therefore the computer uses only 192 lines for one picture.

ANTIC now works synchronously with the generation of the TV picture, using it's own control program, the display list. Now we will take a look at the display list. The program in fig.2 gives us a printout of this.

```
10 A=PEEK(560)+256*PEEK(561)
15 V=1
17 H=8
20 FOR I=A TO A+31
22 POSITION H,V
24 V=V+1:IF V>16 THEN V=1:H=20
30 LPRINT I;"  ";PEEK(I)
40 NEXT I
```

```
39968   112          39984   2
39969   112          39985   2
39970   112          39986   2
39971   66           39987   2
39972   64           39988   2
39973   156          39989   2
39974   2            39990   2
39975   2            39991   2
39976   2            39992   2
39977   2            39993   2
39978   2            39994   2
39979   2            39995   2
39980   2            39996   2
39981   2            39997   65
39982   2            39998   32
39983   2            39999   156
```

Fig. 2

The memory locations 560 ($230) and 561 ($231) contain the starting address of the display list. This is the decimal address 39968 ($9C20). We make a printout of the next 32 locations. This is also shown in fig. 2.

Now, what do these numbers mean? To answer this question, we need a programming sheet for the ANTIC. This is shown in fig. 3. Unfortunately, these are all hex numbers, so we have to change our decimal addresses to hex. 112 ($70) means that we start out with 3*8, or 24, empty lines. The next command is 66 ($42). On the left side of our programming sheet, we find CHR(40,2,8).

This means that we have 40 characters across the screen, with 2 colors, and the height of one character is 8 TV lines. On the top is "LD MEM SCAN". With this command, the starting address of the next buffer has also been loaded. This is the following two bytes: 64 ($ 40) and 192 ($ 8C). This is the address $ 8C40(?). The lower address byte comes first, followed by the higher address byte, as per 6502 standards. $9C40 is the decimal address 40000. The next 23 lines all contain the value 02. This means that CHR (40,2,8) is used for the whole picture. The last command is 65 ($ 41). This is a jump instruction. The address of the JUMP is 32 ($ 20), the starting address of the display list. But this jump is not executed immediately. The processor waits for a vertical gap, when the beam moves from point Z back to point A, as in fig.1. Now let's look at how many TV lines we have. We have 3*8, or 24, blank lines and 24*8, or 192, lines with text.

We will now change the display list. First we will type in SHIFT-CLEAR. This clears the screen. Next we poke 39986,112.
This changes the text line to 8 empty TV lines. If we list our program, this line is not used for text. By hitting the system reset key or by poking 39986,02, we reset the display list.

Let's do another trial. Now we poke 39985,07. A black line appears on screen, twice as high as a normal text line. When we list our program, the letters in this line are also twice as high as normal. 07 means CHR(20,5,16); that is, 20 characters in one line with 5 colors, using 16 TV lines. But the following text is displaced.

We have only 20 characters on this line. The remaining characters are printed on the next line. We get rid of this when we POKE 39984,07. Now we have two lines of text, twice as high as normal, in the middle of our TV screen.

In our next experiment with the display list, we change the starting address of the text buffer by poking 39973,00. On the screen, we now have only garbage. We see different types of graphics characters, some of them changing. These are the contents of memory locations, altered by the program. If we hit a few keys, some of the characters will change, but nothing else happens. By using the system reset key, we get our normal screen back.

HSCROL		XX		XX		XX		XX		XX		XX		XX		XX		XX	Horizontal Scrolling
VSCROL			XX	XX			XX	XX			XX	XX			XX	XX			Vertical Scrolling
LD MEM SCAN					XX	XX	XX	XX					XX	XX	XX	XX			Load memory scan (3 byte)
INST INTERRUPT									XX	XX	XX	XX	XX	XX	XX	XX			Display instruction interrupt
BLK 1	00								80										Blank 1 line
" 2	10								90										Blank 2 lines
" 3-7																			Blank 3 thru 7 lines
" 8	70								F0										Blank 8 lines
JMP	01								81										Jump (3 byte instruction);
JVB	41								C1										Jump & wait for Vert. Blank (also 3 byte)
CHR (40,2,8)	02	12	22	32	42	52	62	72	82	92	A2	B2	C2	D2	E2	F2			
" (40,2,10)	03	13	23	33	43	53	63	73	83	93	A3	B3	C3	D3	E3	F3			
" (40,4,8)	04	14	24	34	44	54	64	74	84	94	A4	B4	C4	D4	E4	F4			Character Mode
" (40,4,16)	05	15	25	35	45	55	65	75	85	95	A5	B5	C5	D5	E5	F5			Instructions
" (20,5,8)	06	16	26	36	46	56	66	76	86	96	A6	B6	C6	D6	E6	F6			
" (20,5,16)	07	17	27	37	47	57	67	77	87	97	A7	B7	C7	D7	E7	F7			
MAP (40,4,8)	08	18	28	38	48	58	68	78	88	98	A8	B8	C8	D8	E8	F8			
" (80,2,4)	09	19	29	39	49	59	69	79	89	99	A9	B9	C9	D9	E9	F9			
" (80,4,4)	0A	1A	2A	3A	4A	5A	6A	7A	8A	9A	AA	BA	CA	DA	EA	FA			
" (160,2,2)	0B	1B	2B	3B	4B	5B	6B	7B	8B	9B	AB	BB	CB	DB	EB	FB			Memory Map Mode
" (160,2,1)	0C	1C	2C	3C	4C	5C	6C	7C	8C	9C	AC	BC	CC	DC	EC	FC			Instructions
" (160,4,2)	0D	1D	2D	3D	4D	5D	6D	7D	8D	9D	AD	BD	CD	DD	ED	FD			
" (160,4,1)	0E	1E	2E	3E	4E	5E	6E	7E	8E	9E	AE	BE	CE	DE	EE	FE			
" (320,2,1)	0F	1F	2F	3F	4F	5F	6F	7F	8F	9F	AF	BF	CF	DF	EF	FF			

Number of TV lines per cell

Number of Colors (Background + Playfield types)

Number of Horizontal cells (standard width screen)

(From ATARI Hardware Manual)

Fig. 3

The memory location 39972 contains the lower byte of the starting address. By poking POKE 39972,65, we shift the text one space to the left, and by poking POKE 39972,62, two places to the right.

```
100 POKE 39973,0
105 FOR J=200 TO 255
110 FOR I=0 TO 255
120 POKE 39972,I
130 NEXT I
140 POKE 39973,J
150 NEXT J
```

Fig. 4

The program in fig. 4 shifts the starting address of the text buffer from $ 00 through $ FF and from $ C800 through FFFF. Different wandering patterns are created on the screen.

For the next task, we will make one text line on the screen disappear. We first erase the screen with SHIFT-CLEAR, then we list the program from fig. 4. If we POKE 39978,34, line number 130 disappears. With POKE 39978,02 it re-appears. In the display list, we have changed the command $02 to $22. This means vertical scrolling (VSCROL). Line 130 is hidden behind line 120.

The program in fig. 5 changes the screen into 12 lines of text with 20 characters each.

```
100 POKE 39971,71
110 FOR I=39974 TO 39985
120 POKE I,7
125 NEXT I
130 POKE 39986,65
140 POKE 39987,32:POKE 39988,156
150 PRINT CHR$(125)
160 PRINT "THIS TEXT IS"
170 PRINT "PRINTED ON THE SCREEN""
180 PRINT "IN CAPITAL LETTERS"    "
```

Fig. 5

By doing this, we define a new display list by poking the appropiate values into the memory locations.

Display List Interrupts and the ATARI

Display List Interrupts and the ATARI

When we have worked with the display list of the ANTIC, we have only used the commands in the left half of the lable, shown in fig. 1. The command in the right half of this table starts on display list interrupt (DLI). Before we deal with these DLI's, we will examine what an interrupt is and what happens, when an interrupt occurs. An interupt is a break in a running program by an external signal. When this signal reaches the CPU, the command, which is being executed is finished. Then the address in the program counter and the content of the status register are placed into the stack. Then the CPU performs an indirect jump to the interrupt service routine.

There are two different kinds of interrupts with a 6502 CPU. There is the NMI, the non-maskable interrupt and the IRQ, the interrupt request. The IRQ can be controlled by program. If the interrupt bit in the status registers are not zero, an IRQ is not accepted (or vice-versa, a zero lets the CPU acknowledges an interrupt). This bit in the status-register is set or reset by the assembler commands SEI or CLI.

The second possibility to interrupt a program in the NMI. This interrupt cannot be controlled by program and is always executed.

The starting address for the interrupt service routines are in location $ FFFE, $ FFFF for the IRQ and $ FFFA, $ FFFB for the NMI. These are memory locations in ROM which cannot be altered.

The interrupt service routine starts in ROM, but then it does an indirect jump over a memory location in RAM. This is the address $2000, $201 hex or 512, 513 decimal for the display list interrupts.

If you want to use this interrupt, this memory location must contain the starting address of your interrupt service routine. You return to the normal program by a RTI instruction.

In our first example we will use the DLI to display the upper and lower half of the screen in two different colors. Therefore we change the content of a color register. The blue hue of the screen in graphics mode 0 is determined by the content of color register 2. The following program in machine code changes the content of the color register. It is written as an interrupt service routine.

```
PHA          48        ; (A) → stack
LDA # DE     A9 DE     ; DE means bright green
STA WSYNC    80 0A DY  ; see text
STA COLOR2   8D 18 D0  ; DE → color rey 2
PLA          68        ; get content of accu back from
                         stack
RTI          40        ; go back to normal program
```

First the content of the accumulator is saved on the stack because it is changed in this program. The command STA WSYNC has a special meaning. While working on this interrupt, the CPU needs 33 machine cycles until it starts our program. In this time the electron beam has nearly crossed the screen. If the content of the color register is changed now, the color will change from blue to green. But the duration from the start of the interrupt and the change of the color register is not always the same. There are changes of a few micro seconds. The border of the color change will jiggle slightly back and forth. Using the STA WSYNC command, the 6502 CPU is set to wait until a horizontal sync occurs. Then the color is changed with STA COLOR2. The code of the color is written directly to the color register of the CTIA and not to the RAM location.
If this is done, the color would be changed on the next vertical blanking and not midway thru the screen's cycle.

The color change shows the next program:

```
10 DLI=PEEK(560)+256*PEEK(561)
20 POKE DLI+16,130
30 FOR I=0 TO 10
40 READ A:POKE 1536+I,A:NEXT I
50 DATA 72,169,222,141,10,212
60 DATA 141,24,208,104,64
70 POKE 512,0:POKE 513,6
80 POKE 54286,192
```

In line 10 the beginning of the display list is encounterd and then a byte of this list is changed from $02 to $82 = 130. That means 40 characters in one line, two colors, height equal to 8 TV lines and starting an interrupt. (See figure).

39968	112	39984	2
39969	112	39985	2
39970	112	39986	2
39971	66	39987	2
39972	64	39988	2
39973	156	39989	2
39974	2	39990	2
39975	2	39991	2
39976	2	39992	2
39977	2	39993	2
39978	2	39994	2
39979	2	39995	2
39980	2	39996	2
39981	2	39997	65
39982	2	39998	32
39983	2	39999	156

The machine language program is poked into memory locations starting at $600. This address is stored in $200, 201 (512, 513 decimal). Then the interrupt is enabled by writing a $C8 = 192 to the interrupt enable register of the CTIA at location $ D40E = 54286.
After starting the program we see a sharp border between blue and green.
If we change line 50 to

50 DATA 72, 169, 222, 234, 234, 234

we replace the STA WSYNC command with three NOP's. Now, the border between blue and green is jiggling on the right side of the screen.
In our next example we will use the display list interrupt several times. But the interrupt service routines mustn't be very long. The change of the color or whatever else you want to do must be finished when the electron beam starts the next TV line. For showing multiple DLI's we use our program "MONSTER" on page 114. We change lines 400 to 490 as it is shown in the next figure.

```
50 PRINT CHR$(125)
60 POSITION 10,2
70 PRINT "ONE MOMENT PLEASE"
100 FOR I=0 TO 1023
110 POKE 16384+I,PEEK(57344+I)
115 POKE 17408+I,PEEK(57344+I)
120 NEXT I
200 RESTORE
205 FOR I=16648 TO 16743
210 READ A
220 POKE I,A
230 NEXT I
250 FOR I=17672 TO 17767
260 READ A
270 POKE I,A
280 NEXT I
400 PRINT CHR$(125)
402 POKE 756,64
405 FOR Y=1 TO 17 STEP 4
410 FOR X=3 TO 35 STEP 5
420 POSITION X,Y
430 PRINT "ABCD"
440 POSITION X,Y+1
450 PRINT "EFGH"
460 POSITION X,Y+2
470 PRINT "IJKL"
475 NEXT X
477 NEXT Y
500 POKE 756,68
510 FOR I=1 TO 50:NEXT I
520 POKE 756,64
530 FOR I=1 TO 50:NEXT I
540 GOTO 500
1000 DATA 0,1,96,48,24,13,3,7
1005 DATA 0,129,139,63,127,255,255,135
1010 DATA 0,129,130,252,254,255,255,225
1015 DATA 0,128,6,12,24,176,161,224
1020 DATA 7,15,15,15,7,7,3,1
1025 DATA 3,3,3,7,207,255,255,255
1030 DATA 192,192,192,224,243,255,255,255
1035 DATA 224,240,240,240,224,224,192,128
1040 DATA 1,1,7,15,15,3,0,0
1045 DATA 255,159,224,248,252,240,0,0
```

```
1050 DATA 255,249,7,31,63,31,0,0
1055 DATA 128,128,224,240,240,192,0,0
2000 DATA 0,1,96,48,24,13,3,7
2005 DATA 0,129,139,63,127,255,255,135
2010 DATA 0,129,130,252,254,255,255,225
2015 DATA 0,128,6,12,24,176,161,224
2020 DATA 7,15,15,15,7,7,3,1
2025 DATA 3,3,3,7,207,255,0,0
2030 DATA 192,192,192,224,243,255,0,0
2035 DATA 224,240,240,240,224,224,192,128
2040 DATA 1,1,7,15,15,3,0,0
2045 DATA 255,159,224,248,252,240,0,0
2050 DATA 255,249,7,31,63,31,0,0
2055 DATA 128,128,224,240,240,192,0,0
```

When we run this program, we have 5 rows of 7 monsters, opening and closing their mouth. Using the DLI, every row will have a different color and the mouth will be open or closed. We use 4 interface service routines of the following form:

```
PHA              48
LDA #COLORNR     A9 DE
STA WSYNC        8D 0A D4
STA COLOR2       8D 18 D0
LDA #CHARSET     A9 44
STA CABAS        8D 09 D4
LDA #ADRELO      A9 15
STA DPIV         8D 00 02
PLA              68
RTI              40
```

First, we load the accumulator with a color code wait for WSYNC and change the color register. Then we change the base address of the character set and set a new start address for the next interrupt service routine.

This is exactly the same routine, except for the colorcode, the base address of the character set and the starting address for the third service routine.

In the fourth service routine the starting address is reset to the beginning of the first routine.

The program shows the next figure.

```
50 PRINT CHR$(125)
60 POSITION 10,2
70 PRINT "ONE MOMENT PLEASE"
100 FOR I=0 TO 1023
110 POKE 16384+I,PEEK(57344+I)
115 POKE 17408+I,PEEK(57344+I)
120 NEXT I
200 RESTORE
205 FOR I=16648 TO 16743
210 READ A
220 POKE I,A
230 NEXT I
250 FOR I=17672 TO 17767
260 READ A
270 POKE I,A
280 NEXT I
400 PRINT CHR$(125)
402 POKE 756,64
405 FOR Y=1 TO 17 STEP 4
410 FOR X=3 TO 35 STEP 5
420 POSITION X,Y
430 PRINT "ABCD"
440 POSITION X,Y+1
450 PRINT "EFGH"
460 POSITION X,Y+2
470 PRINT "IJKL"
475 NEXT X
477 NEXT Y
500 GOSUB 700
510 GOTO 510
520 POKE 756,64
530 FOR I=1 TO 50:NEXT I
540 GOTO 500
700 DPLS=PEEK(560)+256*PEEK(561)
705 POKE DPLS+8,130
710 POKE DPLS+12,130
714 POKE DPLS+16,130
718 POKE DPLS+20,130
720 FOR I=0 TO 83
730 READ A:POKE 1536+I,A:NEXT I
770 POKE 512,0:POKE 513,6
780 POKE 54286,192
```

```
790 RETURN
1000 DATA 0,1,96,48,24,13,3,7
1005 DATA 0,129,139,63,127,255,255,135
1010 DATA 0,129,130,252,254,255,255,225
1015 DATA 0,128,6,12,24,176,161,224
1020 DATA 7,15,15,15,7,7,3,1
1025 DATA 3,3,3,7,207,255,255,255
1030 DATA 192,192,192,224,243,255,255,255
1035 DATA 224,240,240,240,224,224,192,128
1040 DATA 1,1,7,15,15,3,0,0
1045 DATA 255,159,224,248,252,240,0,0
1050 DATA 255,249,7,31,63,31,0,0
1055 DATA 128,128,224,240,240,192,0,0
2000 DATA 0,1,96,48,24,13,3,7
2005 DATA 0,129,139,63,127,255,255,135
2010 DATA 0,129,130,252,254,255,255,225
2015 DATA 0,128,6,12,24,176,161,224
2020 DATA 7,15,15,15,7,7,3,1
2025 DATA 3,3,3,7,207,255,0,0
2030 DATA 192,192,192,224,243,255,0,0
2035 DATA 224,240,240,240,224,224,192,128
2040 DATA 1,1,7,15,15,3,0,0
2045 DATA 255,159,224,248,252,240,0,0
2050 DATA 255,249,7,31,63,31,0,0
2055 DATA 128,128,224,240,240,192,0,0
3000 REM DATA FOR INTERRUPT ROUTINES
3005 DATA 72
3010 DATA 169,222
3015 DATA 141,10,212
3020 DATA 141,24,208
3025 DATA 169,68
3030 DATA 141,09,212
3035 DATA 169,21
3040 DATA 141,00,02
3045 DATA 104,64
3050 DATA 72,169,120,141,10,212
3060 DATA 141,24,208,169,64
3070 DATA 141,09,212,169,42
3080 DATA 141,00,02,104,64
3090 DATA 72,169,180,141,10,212
3100 DATA 141,24,208,169,68
3110 DATA 141,09,212,169,63
3120 DATA 141,00,02,104,64
```

```
3130 DATA 72,169,240,141,10,212
3140 DATA 141,24,208,169,64
3150 DATA 141,09,212,169,00
3160 DATA 141,00,02,104,64
```

For the first time starts the program, by RUN, then by GOTO 200. Do not wait each time for the change of the character sets. The interrupt service routines are poked into the memory in subroutine 700, the data are stored at line 3000 ff.

First all monsters are drawn, then, after a short pause, when the machine code is loaded, every row of monsters is a different color.

It is very interesting to erase program line 402. You see how the text is written to the screen and then changed into the monsters. The uppermost row remains unchanged.

With the display list interrupt, you can divide the screen into horizontal parts. But be careful. It is "critical in time" because it is a real time problem running parallel with the creating of the TV screen.

ATARI 400/800 and CTIA/GTIA

Another video processor in the ATARI is the CTIA, which works closely with ANTIC. It prepares the signals received from the ANTIC for output to the TV set. It adds color, makes the player-missile graphics, and detects collisions. There are nine hardware registers, four of them used for player-missile graphics; the other five, according to the selected text or graphics mode. The memory location of these registers are $D016–$D01A, but they are once more available in the RAM locations $2C4–$ 2C8 (decimal 708–712). During every vertical gap, the contents of these locations are placed in the locations $D016– $ D01 A in the CTIA.

The color itself is determined by a number. With a POKE 712,254, the background color is changed to a bright yellow. The program in fig. 6 changes the hue of the background.

```
10 FOR I=0 TO 254 STEP 2
20 POKE 712,I
30 FOR J=0 TO 25:NEXT J
40 NEXT I
```

Fig. 6

The upper 4 bits of the content of one color register determine the hue; the lower 4 (of which the lowest one is not used), the luminance. Bit pattern %0000 is dark, %1110, light. The number for the hue can be found in table 9.3 of the ATARI BASIC Handbook. The bit pattern %1101 1110 ($ DE or decimal 222) is a bright green. If you POKE 710,222 the background behind the letters will become this color.

The program in fig. 7 shows the background in different colors.

```
10 FOR I=240 TO 254 STEP 2
20 POKE 710,I
30 FOR J=0 TO 25:NEXT J
40 NEXT I
50 FOR I=254 TO 240 STEP -2
60 POKE 710,I
70 FOR J=0 TO 25:NEXT J
```

```
80 NEXT I
90 GOTO 10
```

Fig. 7

The luminance of the letters is determined by color register 709. You may vary only the luminance, not the color, of the letters. If the luminance of the letter and the background are the same, then the letter disappears, as shown in the text program (fig. 8).

```
5 DIM N$(1)
10 POKE 710,222
15 INPUT N$
20 FOR I=0 TO 14
30 POKE 709,I
40 FOR J=0 TO 50:NEXT J
50 NEXT I
60 INPUT N$
70 POKE 709,0
```

Fig. 8

You can use the color register for many purposes. A red background signifies, "pay attention!", or a brightyellow says, "ALL IS WELL". The program in fig. 9 demonstrates this.

```
5 PRINT CHR$(125)
7 DIM N$(1)
10 POSITION 10,10
20 PRINT "K E I N   M  EINGEBEN"
30 INPUT N$
40 IF N$<>"M" THEN GOTO 100
43 POSITION 10,10
44 PRINT "         AUA              "
45 FOR J=1 TO 100
50 A=PEEK(712)
60 POKE 712,PEEK(710)
70 POKE 710,PEEK(709)
80 POKE 709,A
90 NEXT J
99 END
100 PRINT CHR$(125)
110 POSITION 10,10
120 PRINT "D A N K E "
130 END
```

The ATARI 400/800 and its Character Set

The ATARI 400/800 and its Character Set

Each character seen on the screen is represented of an 8 by 8 matrix of little dots. Each dot is represented by one bit in one machine word, one byte. Therefore each character uses 8 bytes for its representation. The following figure shows the bit pattern for the character A. Inside the ATARI, these bit patterns are stored in ROM beginning in memory location $ E000 = 57344. The first character in $ E000 until $ E007 is the space character. There are 126 characters coded and stored from $ E000 to $ E3FF.

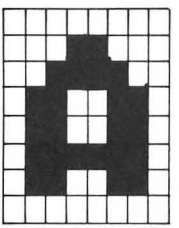
Character	Binary Pattern	Hex.	Dec.
	00000000	0C	0
	00011000	18	24
	00111100	3C	60
	01100110	66	102
	01100110	66	102
	01111110	7E	126
	01100110	66	102
	00000000	00	0

Figure 1

As this character set is in ROM (Read Only Memory) you cannot change it But with the ATARI, it is possible, to define your own character set. The character set must start at a 1K page boundary like, for example, at $ 4000: and the higher address-byte of this boundary must be stored in memory location $2F4 = 756. If you try PRINT PEEK (756) you get 224 and this is $E0.

Now, since we want to play around with this character set, we can use the following program to place the character set in RAM, where we can change it.

```
100 FOR I=0 TO 1023
110 POKE 16384+I, PEEK (57344 + I)
120 NEXT I
130 POKE 756,64
200 FOR I=16684 TO 16655
210 PRINT PEEK(I)
220 NEXT I
```

Figure 2
Move character set into hex 4000

This mini-program also prints the bit pattern for the character A.
To calculate the starting address of the bit pattern of the character A, do the following:
The space character has the decimal code 32 the character A has the decimal code 65; the starting address of the bit pattern for the character A is then

16384 + 33 * 8 = 16648

In line 130 the page boundary is changed from $ E0 to $40 = 64. Now the ATARI uses the character set in RAM which is changeable. With a POKE 16388,255 we change the space character. After this poke, every space character has a bright line in its middle. Only the cursor, the inverted space character, has a dark line.

With POKE 16388,0 we switch back to the normal mode. If we change lines 200 until 220 in the above listing into the lines shown below, the character A is rotating on a horizontal axis. This program can only be stoped by hitting the RESET key. After doing this, the address in memory location 756 is the starting address of the character set in ROM.

```
200 FOR I=16648 TO 16655
205 POKE 16655, PEEK(16648)
210 POKE I-1, PEEK(I)
220 NEXT I
230 GOTO 200
```

Figure 3

You can obtain very different figures by changing the bit pattern of a letter. We will show this in our next program MONSTER in Figure 5. Figure 4 shows the design of our diminitive monster. It is represented by a 4 by 3 matrix of the letters.

A B C D
E F G H
I J K L

Further we will show how easy it is to change the figure by using two different character sets.

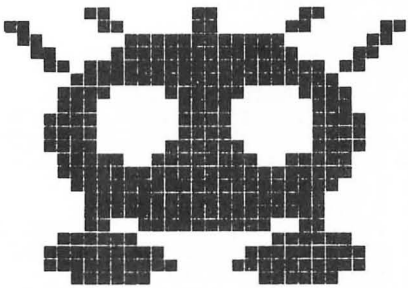


Figure 4

The bit pattern of each letter is changed and converted to a decimal number. These numbers are inserted as DATA statements in the program in Figure 5.
The first character set is defined in the lines 1000 to 1055.
The second character set (which differs only in the letters F and G) is defined in lines 2000 to 2055. In this character set, the two last bytes of the letter F and G are set to zero. This results in a rectangular open mouth on our monster.

The program starts by copying the original character set to $4000 = 16384 and $4400 = 17403. Then the bit pattern for the letters A – L are changed by reading the newly definded values. (Lines 200 – 280). In line 405, the first character set is chosen.

```
100 FOR I=0 TO 1023
110 POKE 16384+I,PEEK(57344+I)
115 POKE 17408+I,PEEK(57344+I)
120 NEXT I
200 RESTORE
205 FOR I=16648 TO 16743
210 READ A
220 POKE I,A
230 NEXT I
250 FOR I=17672 TO 17767
260 READ A
270 POKE I,A
280 NEXT I
400 X=10:Y=10
405 POKE 756,64
410 PRINT CHR$(125):REM CLEAR SCREEN
420 POSITION X,Y
430 PRINT "ABCD"
440 POSITION X,Y+1
450 PRINT "EFGH"
460 POSITION X,Y+2
470 PRINT "IJKL"
500 POKE 756,68
510 FOR I=1 TO 50:NEXT I
520 POKE 756,64
530 FOR I=1 TO 50:NEXT I
540 GOTO 500
1000 DATA 0,1,96,48,24,13,3,7
1005 DATA 0,129,139,63,127,255,255,135
1010 DATA 0,129,130,252,254,255,255,225
1015 DATA 0,128,6,12,24,176,161,224
1020 DATA 7,15,15,15,7,7,3,1
1025 DATA 3,3,3,7,207,255,255,255
1030 DATA 192,192,192,224,243,255,255,255
1035 DATA 224,240,240,240,224,224,192,128
1040 DATA 1,1,7,15,15,3,0,0
1045 DATA 255,159,224,248,252,240,0,0
1050 DATA 255,249,7,31,63,31,0,0
1055 DATA 128,128,224,240,240,192,0,0
2000 DATA 0,1,96,48,24,13,3,7
2005 DATA 0,129,139,63,127,255,255,135
2010 DATA 0,129,130,252,254,255,255,225
2015 DATA 0,128,6,12,24,176,161,224
2020 DATA 7,15,15,15,7,7,3,1
2025 DATA 3,3,3,7,207,255,0,0
2030 DATA 192,192,192,224,243,255,0,0
2035 DATA 224,240,240,240,224,224,192,128
2040 DATA 1,1,7,15,15,3,0,0
2045 DATA 255,159,224,248,252,240,0,0
2050 DATA 255,249,7,31,63,31,0,0
2055 DATA 128,128,224,240,240,192,0,0
```

Figure 5
MONSTER

The print statement PRINT "ABCD" in line 430 doesn' t print the letter, but the newly defined code. That is the first line of our monster. The next lines are printed with the remaining two print statements.
Then in the lines 500 to 540 we switch between two character sets following a short delay loop. Our monster opens and closes its rectangular mouth.

We could use a third character set and define the bit pattern for the letter G as

DATA 255, 255, 255, 255, 255, 255, 255, 255

If we call this character set, our monster closes its left eye.
As a serious application of this technique, you could think of defining the shapes of electronic symbol and use them for the design of electronic circuits.

ELCOMP

BOOKS and SOFTWARE

For ATARI – PET – OSI – APPLE II – 6502

ATARI BASIC – Learning by Using
This new book is an "Action"-Book. You do more than read it. Learn the intricacy of ATARI-BASIC thorugh the short programs which are provided. The suggestions challenge you to change and write program routines. Yes, it's exciting –
Many of the programs are appropriate for beginners as well as experienced computer users. (Screen Drawings, Special Sounds, Keys, Paddles + Joysticks. Specialized Screen Routines, Graphics and Sound. Peeks and Pokes and special stuff).
Order-No. 164 **$9.95**

Games for the ATARI-Computer
How to program your own games on the ATARI. Complete listings in BASIC and Machine Language of exciting games. Tricks and hints.
Order-No. 162 **$7.95**

ATMONA-1
Machine Language Monitor for the ATARI 400/800.
This powerful monitor provides you with the firmware support that you need to get the most out of your powerful system. ATMONA-1 comes on a bootable cassette. No cartridges required. Disassemble, Memory Dump HEX + ASCII, (Change Memory Locations, Blocktransfer, fill memory block, save and load machine language programs, start mach. Lang. Progr. (Printer optional).
Comes with introductionary article on how to program the ATARI computer in machine language. (Available also in ROM)
Order-No. 7022 **$19.95**

ATMONA-2 Superstepper
A very powerful Tracer to explore the ATARI ROM/RAM area. Stop at previously selected address. Opcode or operand (cassette).
Order-No. 7049 **$49.95**

The Third Book of Ohio Scientific is now available!
Very important information for the OSI system experimenter. Interface techniques, system expansions, accessories and much more (EPROM-Burner, 6522 I/O-card with 1K RAM, Soundboard, EPROM/RAM board).
Order-No. 159 **$7.95**

The Fourth Book of OHIO
VIP-Book – Very Important Programs. Many interesting programs for OSI computers. Sorting (Binary Tree). Differential Equitation, Statistics, Astrology, Gas Consumption, Games a. s. o.
Order-No. 160 **$9.95**

VIP Package – Above book plus a cassette with the programs.
Order-No. 160 A **$19.95**

The Fifth book of Ohio Scientific
Many exciting programs programming hints and tricks, Textwriter, Debugger for C1P, Games, Utilities and much more (polled keyboard)
Order-No. 161 **$7.95**

Invoice Writing Program for OSI-C1PMF, C4P. Disk and Cassette, 8K RAM.
Order-No. 8234 **$29.80**

Mailing List for C1PMF or C4PMF 24K RAM
250 addresses incl. phone number and parameters on one 5 1/4 disk)
Order-No. 8240 **$29.80**

8K Microsoft BASIC Reference Manual
Authoritative reference for the original Microsoft 4K + 8K BASIC developed for ALTAIR and later computers including OSI, PET, TRS-80 and VIC.
Order-No. 141 **$9.95**

Expansion Handbook for 6502 and 6802
S-44 Card Manual describes all of the 4.5 x 6.5 44-pin S-44 cards incl. schematics. A MUST for every 6502 system user (KIM, SYM, AIM, VIC, PET, OSI)
Order-No. 152 **$9.95**

EDITOR/ASSEMBLER for ATARI 800, 32K RAM
Extremely fast and powerful Editor/Assembler. (8K Source-code in about 5 seconds) Includes ATMONA-1.
Order-No. 7098 **$49.95**

MACRO-Assembler
for ATARI 800, 48K RAM
Please specify your system: RAM, disc or cassette.
Order-No. 7099 **$89.00**

Gunfight – For ATARI 400/800 16K RAM, needs two joysticks, animation and sound. (8K machine language).
Order-No. 7207 **$19.95**

EPROM BURNER for ATARI 400/800. Bare boards only with description, schematic + software (2716, 2732).
Order-No. 7041 **$99.00**

Invoice Writing for very small business with ATARI 400/800 16K RAM.
Order-No. 7022, cass. **$29.85**
Order-No. 7200, disc. **$39.99**

Wordprocessor for ATARI 800, 48K RAM
Order-No. 7210 **$29.95**

How to connect your EPSON-Printer to the ATARI 400/800. Construction article with printed circuit board and software. (Screenprint and variable characters per line).
Order-No. 7210 **$19.95**

OSI OSI OSI OSI OSI

The First Book of Ohio Scientific
Introduction to OSI computers. Diagrams, hardware and software information not previously available in one compact source. 192 pages.
Order-No. 157 **$7.95**

The Second Book of Ohio Scientific
Very valuable information about OSI microcomputer systems. Introduction to OS-65 D and OS-65U networking. Hardware and software hints and tips. Systems specifications. Business applications.
Order-No. 158 **$7.95**

Microcomputer Application Notes
Reprint of Intel's most important application notes including 2708, 8085, 8255, 6251 chips. Very necessary for the hardware buff.
Order-No. 153 **$9.95**

Complex Sound Generation
New revised applications manual for the Texas Instruments SN 76477 Complex Sound Generator.
Order-No. 154 **$6.95**

Small Business Programs
Complete listings for the business user. Inventory, Invoice Writing, Mailing List and much more. Introduction to Business Applications.
Order-No. 156 **$14.90**

Microcomputer Hardware Handbook
Descritions, pinouts and specifications of the most popular microprocessor and support chips. A MUST for the hardware buff.
Order-No. 29 **$14.95**

Care and Feeding of the Commodore PET
Eight chapters exploring PET hardware. Includes repair and interfacing information. Programming tricks and schematics.
Order-No. 150 **$9.95**

Prototype-Expansion Board for VIC-20 (S-44-Bus).
Order-No. 4844 **$18.95**

16K RAM/ROM board for S44-bus. Any combination of RAM and ROM on one board. (SY2128 or 2716)
Order-N0. 613 **$39.95**

Low cost expanison boards for your APPLE II. Bare board comes with extensive description and software.
Prototyping card
Order-No. 604 **$29.00**
6522 VIA–I/O Exp.
Order-No. 605 **$39.00**
2716 EPROM-Burner
Order-No. 607 **$49,00**
8K EPROM/RAM Card
Order-No. 609 **$29.00**

ELCOMP Publishing, Inc.,
Postbox 1194 Pomona, CA 91769

Payment: Check, Money Order, **VISA**, Mastercharge, Eurocheck. POSTPAID or PREPAID in USA. $5.00 handling fee for C.O.D. All orders outside USA: Add 15 % shipping. CA add 6.5 % sales tax.
ATARI is a registered trademark of ATARI Inc.
APPLE II is a registered trademark of APPLE Inc.

NOTES

NOTES

NOTES

NOTES

NOTES

ATMONA-1

A T M O N A – 1

ATMONA-1 – The new machine language monitor for the ATARI 400 and ATARI 800 computer. ATMONA-1 has been developed to provide you with the firmware support that you need to get the most out of your power system.

ATMONA-1 is available on cassette and provides you with the following features:
Commands:

- D: Disassemble (P)
- M: Memory Dump with or without ASCII Dump (P)
- C: Change content of memory location
- F: Fill memory block with a byte
- B: Transfer memory block
- L: Load a memory block from tape
- S: Save a memory block to tape
- 6: · GOTO, START A PROGRAM

X = RETURN TO ATMONA1
SYSTEM RESET = RETURN TO ATMONA1

ATMONA has very user-friendly dialogs.

(P) = with printer option

```
0800:   00          BRK
0801:   85 C9       STA   $C9
0803:   A9 03       LDA   £$03
0805:   85 CB       STA   $CB
0807:   A9 20       LDA   £$20
0809:   20 65 0C    JSR   $0C65
080C:   C6 CB       DEC   $CB
080E:   D0 F7       BNE   $0807
0810:   C6 C9       DEC   $C9
0812:   10 EF       BPL   $0803
0814:   A9 08       LDA   £$08
0816:   85 C9       STA   $C9
0818:   A0 00       LDY   £$00
081A:   B1 C5       LDA   ($C5),Y
081C:   24 CA       BIT   $CA
081E:   30 16       BMI   $0836
0820:   29 7F       AND   £$7F
0822:   C9 7D       CMP   £$7D
0824:   90 02       BCC   $0828
0826:   A9 A0       LDA   £$A0
```

```
0800    0085C9A90385CBA9    )EI)CEK)
0808    2020650CC6CBD0F7      eLFKPw
0810    C6C910EFA90885C9    FIPo)HEI
0818    A000B1C524CA3016     )1E$JOV
0820    297FC97D9002A9A0    ) I PB)
0828    C920B00209402065    I OBI) e
0830    0C189005109E2075    LXPEP^ u
```

```
0830    0C 18 90 05 10 9E 20 75
0838    08 E6 C5 D0 03 E6 C6 D8
0840    A5 C5 C5 C2 A5 C6 E5 C3
0848    90 17 24 C4 10 21 24 CA
0850    10 1D C6 C9 E6 CA A5 C7
0858    85 C5 A5 C8 85 C6 18 90
0860    A2 C6 C9 D0 B3 24 C4 10
```

ATMONA-II contains a super-duper tracer including a relocator, breakpoints, and register dump.

ATMONA-II provides you with the following features:

1. The Supertracer can stop at a previously selected address
2. The Supertracer can stop at a previously selected operand
3. The Supertracer can stop at a previously selected Op-Code
4. The Supertracer stops at every BREAK-command, RTI-command, set-interrupt-command and at every Op-Code which does not represent a 6502-instruction.

Pricing information:

ATMONA-1	Order-No. 7022	$19.95
ATMONA-2	Order-No. 7049	$49.95

Ing. W. Hofacker GmbH, 53 Redrock Lane, Pomona, CA 91766 / USA, (714) 623 8314

Ing. W. Hofacker GmbH, Tegernseerstr. 18, D-8150 Holzkirchen / West Germany

This is preliminary information.

ATARI is a trademark of ATARI Inc., Sunnyvale, California

HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER